Trabajo de grado:
"*n* veces a través del espejo"

Cecilia Pertino
Universidad Nacional de La Plata
La Plata, Argentina

Director: Dr. Juan V. Echagüe

Resumen

Este trabajo se enfoca en el área de *semántica de sistemas reactivos* que, a diferencia de los sistemas relacionales o funcionales, se caracterizan por mantener una interacción con el ambiente. Sobre dichos sitemas se define una gran clase de *operadores unarios*, cuyos representantes típicos son los operadores de prioridad, de renombrado y de ocultamiento y se estudia su compatibilidad con respecto a equivalencias semánticas en modelos de *sistemas de transiciones etiquetadas* (STE).

En principio se consideran los operadores de prioridad y se obtiene que la *n-nested simulación* [GV89] no es una congruencia para ellos. Se muestra que la *mayor congruencia* contenida en esta equivalencia es estrictamente más gruesa que la $n+1$-nested simulación.

Luego se generaliza el conjunto de operadores para incluir muchos de los operadores unarios estáticos usados en sistemas reactivos y se obtiene que *ready-simulación* [BIM88, Gla90] y *bisimulación* [Par81] son congruencias para ese conjunto. Se demuestra además que la *n-nested simulación* es una congruencia sólo cuando el operador unario es monótono.

Finalmente se introduce una nueva familia infinita de equivalencias semánticas sobre los STE, la *n-nested ready simulación* que posee las propiedades requeridas de congruencia para todos los operadores de la clase estudiada. Esta familia es estrictamente más fina que la mayor congruencia para estos operadores que está contenida en la n-nested simulación.

Capítulo 1

Introducción

En este trabajo nos interesamos en la *semántica de sistemas reactivos* [Pnu85]. Estos sistemas, cuyos ejemplos clásicos son los sistemas de control industrial, manejadores de redes de computadoras y componentes en general de sistemas operativos, se caracterizan principalmente por el hecho de mantener una interacción con el ambiente.

No es adecuado representar estos programas por medio de una función o una relación a ser calculada, como es el caso en otros sistemas que por ello reciben el nombre de *sistemas funcionales* o *relacionales*. Ejemplos de éstos son los programas de cálculo numérico, compiladores y todos aquellos sistemas en los cuales la interacción con el ambiente pueda verse como un único punto de entrada de datos y otro de salida. Más aún, como no calculan funciones, no es relevante que un sistema reactivo termine: los sistemas operativos, por ejemplo, nunca lo hacen.

Los modelos más usuales de sistemas reactivos están basados en *sistemas de transiciones etiquetadas (STE)* [Kel76]. Estos son grafos dirigidos en donde los nodos representan estados del sistema y las aristas las transiciones entre ellos, etiquetadas por acciones visibles desde el exterior en las cuales se basa la interacción del sistema con el ambiente.

La popularidad de los STE se debe a su simplicidad, su clara representación gráfica, facilidad de definición de las operaciones que representan los constructores usuales de los sistemas reactivos (composición alternativa, composición paralela, comunicación y prioridad entre otras) y la existencia de una técnica simple, llamada *Semántica Operacional Estructurada* [Plo81], para dar semántica sobre los STE a lenguajes de programación (no necesariamente reactivos) [Hen90].

En la literatura aparecen muchos lenguajes simples, llamados *lenguajes de descripción de procesos*, propuestos para representar sistemas reactivos. Por ejemplo CCS [Mil80, Mil89], CSP [BHR84, Hoa85] y ACP [BK85, BW90], los cuales poseen semánticas basadas en STE.

En general los STE no pueden ser usados directamente para dar semántica a un lenguaje porque son demasiado concretos. Esto es, diferencian sistemas que consideramos deberían ser equivalentes. Por ejemplo hacen distinción entre los sistemas a y $a + a$, representados en la figura 1.1, que, no se puede negar, tienen el mismo comportamiento: ambos comienzan ejecutando una acción a y terminan inmediatamente después. Esta idea de "igual comportamiento" es la que se usa para dar semántica a los lenguajes. Hay numerosas formas de definir que dos sistemas tienen el mismo comportamiento, lo que da lugar a las distintas relaciones de

equivalencia entre sistemas, que llamamos *equivalencias semánticas*.

Figura 1.1: Ejemplo de procesos con el mismo comportamiento

Entonces, para tratar con el problema de bajo nivel de abstracción de los STE, se da la semántica en dos pasos. Primero se asocia el sistema a un STE (su semántica concreta) y luego se considera el conjunto de STE módulo una equivalencia semántica. De esta manera asociamos cada sistema a una clase de equivalencia de los STE: su semántica abstracta. Se han propuesto numerosas equivalencias semánticas en la literatura. Entre las más populares se encuentran la equivalencia por trazas [OH83], la failure equivalencia [BHR84] y la bisimulación [Par81]. Un extenso estudio comparativo de estas equivalencias puede encontrarse en [Gla90] y en [Gla93].

La semántica en dos pasos exige cierto cuidado en el momento de definir los operadores y las equivalencias a ser usados. Esto se debe a que, en general, los constructores de los lenguajes se describen como operaciones sobre los STE. Para cada equivalencia semántica es necesario considerar uno a uno los constructores definidos y verificar si continúan teniendo sentido en las clases de equivalencia obtenidas. En otras palabras, si tenemos en el STE un operador α que representa adecuadamente cierta construcción de un lenguaje y estamos interesados en usar la equivalencia semántica $\sim$. Es posible elevar la operación α definida sobre el STE a la clase de equivalencia obtenida a partir de $\sim$?

Para verificarlo es suficiente probar que la equivalencia $\sim$ es una *congruencia* para el operador α, esto es: si $\mathcal{G}_1 \sim \mathcal{G}_2$ entonces $\alpha(\mathcal{G}_1) \sim \alpha(\mathcal{G}_2)$. Esta propiedad también se conoce como *compatibilidad* del operador con la equivalencia, e intuitivamente nos dice que si dos STE pertencen a la misma clase de equivalencia sus imágenes a través del operador permanecen juntas. En otras palabras, dos sistemas equivalentes deben permanecer equivalentes después de la aplicación del operador.

Como podemos ver, que la equivalencia elegida sea una congruencia para las operaciones consideradas es una condición natural y necesaria para el uso de la misma como equivalencia semántica para los STE en un cierto contexto.

Dentro del marco de los STE encontramos una clase de operadores interesante, la clase de los operadores de prioridad. La aplicación de un operador de prioridad a un STE consiste en la eliminación de las opciones de "baja prioridad" en el momento en que el sistema debe elegir entre varias alternativas. En las aplicaciones reales es imprescindible usar operaciones de este tipo.

Los operadores de prioridad aparecieron por primera vez en la literatura de sistemas reactivos en [BBK86]. Formalmente un operador de prioridad está asociado a un orden parcial $\geq$, el orden de prioridad sobre el conjunto de acciones. Cuando se aplica a un STE, se

consideran las etiquetas de las transiciones salientes en cada estado y se borran aquellas transiciones cuyas etiquetas no son $\geq$-maximales.

Una equivalencia semántica particularmente interesante es la bisimulación [Par81]. Intuitivamente dos sistemas son bisimilares si pueden participar exitosamente de un refinado juego de imitación (en un sentido muy preciso) que permite a los participantes intercambiar, en cualquier momento del juego y cualquier cantidad de veces, los roles de imitado e imitador. Un proceso propone la realización de una acción y el otro lo imita. Luego pueden cambiar de rol y continuar con la imitación.

Se sabe que la bisimulación es una congruencia para los operadores de prioridad [BBK85, BBK86]. También que la equivalencia por trazas y la failure equivalencia no lo son [BBK85] y que la congruencia mas gruesa para los operadores de prioridad contenida en éstas equivalencias es la ready-trace [BBK85].

Las n-nested simulaciones (para $n = 1, 2, \ldots$), que notamos $\rightleftharpoons_n$, forman una familia infinita de equivalencias distintas que son estrictamente más finas cuando n crece. Se extienden desde la simulación, coincidiendo con la 1-nested simulación, hasta la bisimulación: Dos STE de bifurcación finita son bisimilares solamente si son n-nested similares para un n lo suficientemente grande.

Dos sistemas son n-nested similares cuando pueden participar exitosamente del mismo juego de la bisimulación cuando el número de intercambio de roles está limitado a un máximo de $n - 1$.

De la misma manera que la bisimulación y otras equivalencias , la n-nested simulación posee una caracterización lógica muy simple [HM85].

Finalmente, es importante el papel que juega la 2-nested simulación en la teoría de Semántica Operacional Estructurada. En [GV89] se muestra que esta equivalencia es la congruencia más gruesa, contenida en la equivalencia de trazas completas, con respecto a una gran clase de operadores: los que se definen en *formato tyft/tyxt*.

1.1 Contribución

En éste trabajo se realiza un estudio más extensivo en cuanto a la congruencia de las distintas equivalencias semánticas encontradas en la literatura con respecto a los operadores de prioridad.

Se tienen en cuenta especialmente los resultados con respecto a la equivalencia ready-simulación [BIM88, Gla90], que no aparecen en la literatura y también de una familia de equivalencias, las n-nested simulaciones [GV89], cuyos resultados incitaron a un estudio más profundo.

Ya dijimos que la equivalencia ready-trace es una congruencia para los operadores de prioridad [BBK85]. Una equivalencia más fina que ésta es la ready simulación. Tal como era de esperarse, se muestra que la ready simulación es una congruencia para ellos. Lo que resultó completamente sorprendente es que las n-nested simulaciones no sean congruencias para esos operadores. Esto fue inesperado ya que, a partir de $n = 2$, éstas equivalencias son más finas que la ready-simulación y por lo tanto "disponen de toda la información necesaria" para comportarse correctamente con respecto a la prioridad.

De los resultados anteriores pueden hacerse algunas observaciones inmediatas. La primera es que las $\rightleftharpoons_n$ no pueden ser consideradas equivalencias semánticas, al menos en ninguno de los contextos en que la prioridad tenga sentido. Lo mismo sucede con la simulación, que es un caso particular de n-nested simulación, cuando $n = 1$. La última está relacionada con la Semántica Operacional Estructurada: el hecho de que los operadores de prioridad no sean compatibles con la 2-nested simulación significa que éstos operadores no pueden ser representados en *formato tyft/tyxt*.

A partir de esos resultados es natural preguntarse sobre las equivalencias más gruesas contenidas en las $\rightleftharpoons_n$ que sean congruencias para los operadores de prioridad. Aquí se demuestra que forman una nueva familia, que denominamos *n-nested congruencias* (con respecto a la prioridad) y se intercalan entre la n-nested y la $n + 1$-nested simulación.

También parece natural intentar caracterizar alguna familia de operadores que no sean compatibles con la n-nested simulación. Esto puede arrojar luz sobre el porqué de esta incompatibilidad. En este trabajo se define una clase extensa de operadores que abarcan muchos de los que se usan en los STE. Estos operadores se caracterizan por preservar los estados y las acciones y modificar solamente la relación de transición, eliminando o re-etiquetando transiciones. Una transición saliente de un estado se conserva o no (ocasionalmente con otra etiqueta) dependiendo de las etiquetas de las demás transiciones salientes. Esta nueva clase incluye operadores tradicionales en la literatura tales como los de renombrado [Hoa85, Mil89], encapsulamiento [Mil89, BW90], prioridad [BBK86, BW90] y otros. Los mismos, junto con el producto cartesiano de STE, se pueden usar para representar los distintos operadores de comunicación que aparecen en la literatura, incluyendo los que se proponen en CCS, ACP y CSP, el operador de broadcasting $\bowtie$ [Pnu85] y el propuesto en [CH88, CN94].

Dado que los operadores de prioridad pertenecen a esta clase, no todos ellos son compatibles con $\rightleftharpoons_n$. La pregunta es entonces: cuales son, exactamente, aquellos operadores para los que $\rightleftharpoons_n$ es una congruencia? Y para toda la clase, cuál es la congruencia más gruesa contenida en $\rightleftharpoons_n$?

En este trabajo se demuestra que los operadores compatibles con las $\rightleftharpoons_n$ son exactamente los "monótonos", que son por ello los únicos que podemos escribir en formato *tyft/tyxt*. De aquí se deduce que se pueden usar las n-nested simulaciones en modelos que contengan solamente los operadores monótonos.

Se define entonces una nueva *n-nested congruencia* (con respecto a esta clase de operadores) como la congruencia más gruesa para todos los operadores de la clase contenida en las $\rightleftharpoons_n$. Se prueba que esas equivalencias se encuentran intercaladas entre la n-nested y la $n + 1$-nested simulación.

En la búsqueda de equivalencias para la clase de operadores se introduce finalmente otra familia infinita de equivalencias semánticas, las *n-nested ready simulaciones*, que son variantes de las n-nested simulaciones. Se da su caracterización y se demuestra que son congruencias para los operadores definidos. Finalmente se muestra que la n-nested ready simulación no coincide con la n-nested congruencia sino que es estrictamente más gruesa que ella y estrictamente más fina que la $n + 1$-nested simulación. Esta nueva equivalencia semántica puede entonces usarse en modelos en que los operadores definidos tienen sentido.

1.2 Los resultados

Los resultados nuevos de este trabajo son:

1. Ready simulación es una congruencia para los operadores de prioridad.
2. n-nested simulación no es una congruencia para los operadores de prioridad.
3. Al aplicar una serie de operadores de prioridad a dos STE n-nested similares se obtienen STE $n-1$-nested similares.
4. La mayor congruencia para los operadores de prioridad contenida en $\rightleftharpoons_n$ es estrictamente más fina que $\rightleftharpoons_n$ y estrictamente más gruesa que $\rightleftharpoons_{n+1}$.
5. La clase **U** de operadores definidos es compatible con la ready simulación.
6. Bisimulación es una congruencia para la clase **U**.
7. n-nested simulación no es una congruencia para **U**.
8. La mayor congruencia para la clase **U** contenida en $\rightleftharpoons_n$ es estrictamente más fina que la mayor congruencia para la prioridad y estrictamente más gruesa que la $\rightleftharpoons_{n+1}$.
9. $\rightleftharpoons_n$ es una congruencia para los operadores de **U** sii éstos son monótonos.
10. El resultado anterior incluye a las simulaciones.
11. Se define y caracteriza la n-nested ready simulación ($\rightleftharpoons_{R,n}$).
12. $\rightleftharpoons_{R,n}$ es una congruencia para los operadores de **U**.
13. La $\rightleftharpoons_{R,n}$ es estrictamente más fina que la mayor congruencia contenida en $\rightleftharpoons_n$ y estrictamente más gruesa que $\rightleftharpoons_{n+1}$.

Nota 1.2.1 *De éstos resultados, los primeros 4 fueron publicados en [DEP94b] y los 9 siguientes en [DEP94a].*

1.3 Estructura del trabajo

El trabajo está organizado de la siguiente manera:

El capítulo 2 esta dedicado a las nociones básicas y la presentación de la notación. Allí se definen los *Sistemas de Transiciones Etiquetadas*, junto con las *equivalencias semánticas*. Se presenta además una caracterización alternativa de las equivalencias en la *lógica HML*, de Hennessy y Milner.

En el capítulo 3 se introduce la definición formal de *congruencia* y se presenta la clase de los *operadores de prioridad*. Se analiza la *compatibilidad* de los mismos con cada una de las equivalencias semánticas definidas en capítulo anterior. Se presentan resultados conocidos junto con los nuevos. No se tratan las *n-nested simulaciones* por presentar numerosas diferencias con éstos casos.

La $\rightleftharpoons_n$ se trata en el capítulo 4, en el que se demuestra que la equivalencia no es una congruencia para la clase de operadores de prioridad. Se acota el alcance de éste comportamiento "indebido" de las n-nested simulaciones y se define la *mayor congruencia para la clase de operadores de prioridad* incluida en las n-nested simulaciones. Para concluir se da la ubicación de la nueva equivalencia con respecto a las equivalencias semánticas conocidas.

La definición de la *clase general de operadores*, la clase U, se da en el capítulo 5. Allí se verifica que los nuevos operadores son *compatibles con la ready-simulación y con la bisimulación*. Se demuestra también que los mismos *no son compatibles con* $\rightleftharpoons_n$ y se caracteriza exactamente el subconjunto de operadores de U que sí los son. Estos resultados se transfieren a la *simulación* que ya se dijo es un caso particular de n-nested simulación. Finalmente se define la *mayor congruencia* para esta clase de operadores contenida en las n-nested simulaciones y se da su ubicación con respecto a las demás equivalencias semánticas.

En el capítulo 6 se presenta una nueva familia infinita de equivalencias semánticas: las *n-nested ready simulaciones*. Se demuestra que son congruencias para la clase U y se da el ordenamiento final de las equivalencias semánticas en donde se incluyen todas las definidas en este trabajo.

Se concluye con el capítulo 7 presentando algunas consecuencias de estos resultados y posibilidades de trabajo futuro.

Capítulo 2

Nociones de base

En éste capítulo se da la noción de Sistemas de Transiciones Etiquetadas y se presenta la sintaxis del lenguaje utilizado. Se definen las equivalencias semánticas que se usan a lo largo del trabajo y se las caracteriza según la lógica de Hennessy y Milner.

2.1 Sistemas de Transiciones Etiquetadas

En este trabajo nos interesamos en el comportamiento de los sistemas reactivos. Estos, a diferencia de los programas funcionales o relacionales, se caracterizan por mantener una constante interacción con el ambiente. La interacción con el ambiente se puede definir de diferentes maneras, pero básicamente consiste en las acciones del sistema que se pueden percibir del exterior. Por eso es que no se pueden representar con una función o una relación que reflejan una única entrada y una única salida de datos y en dónde todo el cálculo se realiza internamente.

Se necesita una estructura que permita representar los estados del sistema y las acciones (visibles) que llevan de un estado a otro. Para ello tomamos un modelo muy utilizado en la literatura: los Sistemas de Transiciones Etiquetadas (STE) [Kel76].

Los STE son grafos dirigidos donde los nodos representan los estados de sistema y las aristas las transiciones entre ellos. Estas transiciones están acompañadas por acciones visibles desde el exterior.

Definición 2.1.1 *Un* sistema de transiciones etiquetadas *(STE) es una estructura* $\mathcal{G} = (S, i, Act, \longrightarrow)$ *en donde* S *es un conjunto de* estados *con* estado inicial i, Act *es un* alfabeto de acciones *y* $\longrightarrow \subseteq S \times Act \times S$ *es una* relación de transición. *Los elementos* $(p, a, q) \in \longrightarrow$, *que notamos* $p \xrightarrow{a} q$, *son las* transiciones.

El conjunto de *acciones iniciales* de un estado $p \in S$, en un STE $\mathcal{G}$, está definido por: $I_{\mathcal{G}}(p) = \{a \in Act : \exists q \in S \text{ tq } p \xrightarrow{a} q\}$. Siempre que quede claro en el contexto usamos $I(p)$ en lugar de $I_{\mathcal{G}}(p)$. Decimos que un STE tiene *imagen finita* si para todo $p \in S$ y $a \in Act$ el conjunto $\{q \in S : p \xrightarrow{a} q\}$ es finito. En este trabajo consideramos solamente STE de imagen finita que denotamos $\mathcal{C}_{STE}$.

Un STE como grafo dirigido puede tener ciclos pero en este trabajo sólo se consideran los de ramificación finita.

Sea $\sigma \in Act^*$ una secuencia de acciones (donde ϵ es la secuencia nula). Definimos $\longrightarrow^* \subseteq S \times Act^* \times S$ como la clausura reflexo-transitiva de $\longrightarrow$, es decir,

- $p \xrightarrow{\epsilon} p$ para todo estado p
- si $p \xrightarrow{a} q$ y $q \xrightarrow{\sigma}{}^{\star} r$ entonces $p \xrightarrow{a\sigma}{}^{\star} r$

En general escribimos $p \xrightarrow{\sigma} q$ en lugar de $p \xrightarrow{\sigma}{}^{\star} q$.

2.2 El lenguaje

A fin de poder presentar fácilmente ejemplos para los STE necesitamos representarlos sintácticamente. Para ello utilizamos un pequeño lenguaje de descripción de procesos muy simple, tomado de CCS [Mil80, Mil89]. El mismo nos permite dar una expresión sintáctica a sólo una parte de $\mathcal{C}_{STE}$, exactamente a los árboles finitos, suficiente en este caso.

Definición 2.2.1 *Sean Act un alfabeto de acciones y $a \in Act$. Consideramos el* lenguaje $\mathcal{L}$ *definido por:*

$$s ::= 0 \mid a.s \mid s + s$$

También llamamos *procesos* a los términos del lenguaje.

Cada término o proceso representa un árbol. 0 representa el árbol formado por el nodo raíz que no contiene ningún subárbol. En general escribimos a en vez de $a.0$.

Dados los procesos s y t representamos $a.s$ y $s + t$ como en la figura 2.1.

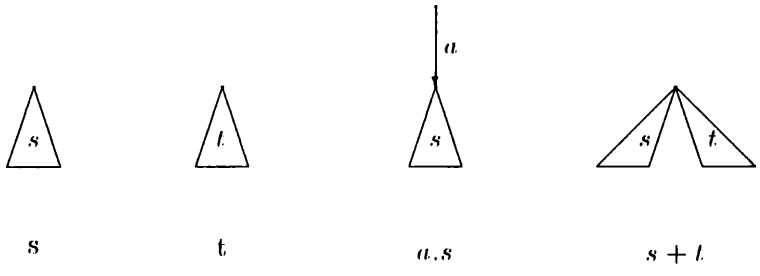


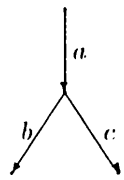
Figura 2.1: Construcción de procesos

Llamamos *acciones atómicas* (o pasos) a las constantes $a, b, c, \ldots$ de Act. Cada uno de los procesos del lenguaje se corresponde con un STE. Intuitivamente se ve que cada elemento del lenguaje es a su vez un *proceso*. En particular 0 corresponde al STE $(\{i\}, i, Act, \longrightarrow)$, en donde $I(i) = \emptyset$. Con ese proceso representamos la inacción.

El proceso que vemos en la figura 2.2 corresponde al STE $(\{a.(b + c), b + c, 0\}, a.(b + c), \{a, b, c\}, \{a.(b + c) \xrightarrow{a} b + c, b + c \xrightarrow{b} 0, b + c \xrightarrow{c} 0\})$.

2.3 Las equivalencias semánticas

Dijimos que los STE no se pueden usar directamente para dar semántica al lenguaje porque son demasiado concretos. Diferencian procesos como a y $a + a$ (figura 1.1) que se comportan



$a.(b + c)$

Figura 2.2: Ejemplo de proceso

de la misma manera: ambos comienzan ejecutando una acción a y terminan inmediatamente después.

Siguiendo [Mil80] para dar semántica a un lenguaje basta con definir una relación de equivalencia entre los procesos o sistemas, identificando aquellos como a y $a + a$ que tienen el mismo comportamiento. En este marco, el "significado" o "la semántica" de un sistema está dado por la clase de equivalencia a la que pertenece, que representa su comportamiento. Por eso decimos que estas relaciones de equivalencia son semánticas.

En la literatura encontramos muchas equivalencias semánticas [GV89, OH83, BHR84, Hoa85, Pnu85, BIM88, Par81, Gla90] definidas para procesos secuenciales de ramificación finita. Estas equivalencias surgen de la necesidad de realizar diferencias más o menos finas según el caso, dependen así del detalle con que se miran los sistemas. Las equivalencias más "gruesas" se fijan solamente en las secuencias de acciones que puede realizar un proceso y las más "finas" examinan detalladamente la secuencia de sus elecciones no determinísticas. Como toda familia de relaciones de equivalencia se la puede ordenar por la relación parcial "hace estrictamente más identificaciones que" o "determina clases de equivalencia más pequeñas que", siendo la equivalencia por trazas [Hoa85] la más gruesa (la que hace más identificaciones) y la bisimulación [Mil83] la equivalencia más fina (la que hace menos identificaciones). De esta manera se forma el reticulado de equivalencias de la figura 2.3 (ver [Gla90]), en donde $\sim_1 \longrightarrow \sim_2$ si $\sim_1$ hace al menos tantas identificaciones como $\sim_2$. Vamos a dedicar el resto de la sección a la definición de éstas equivalencias, comenzando por las más gruesas.

Una traza de un STE es una secuencia de acciones que éste puede realizar, sin tener en cuenta los estados. Decimos que dos STE son equivalentes por trazas si ambos permiten observar el mismo conjunto de trazas.

Definición 2.3.1 (Trazas) *La secuencia de acciones $\sigma \in Act^*$ es una* traza *del STE $\mathcal{G} = (S, i, Act, \longrightarrow)$ si existe $p \in S$ tal que $i \xrightarrow{\sigma} p$. Llamamos $T(\mathcal{G})$ al conjunto de trazas de $\mathcal{G}$. Dos STE, $\mathcal{G}_1, \mathcal{G}_2$ son* equivalentes por trazas *($\mathcal{G}_1 \equiv_T \mathcal{G}_2$) sii $T(\mathcal{G}_1) = T(\mathcal{G}_2)$.*

Trabajar en un modelo con semántica por trazas significa considerar indistinguibles a dos STE que tienen el mismo conjunto de trazas. En la figura 2.4 se da un ejemplo de STE equivalentes por traza: $T(\mathcal{G}_1) = \{a, ab\} = T(\mathcal{G}_2)$.

Esta equivalencia se puede refinar exigiendo que se consideren solamente las trazas completas: aquellas que llevan a estados p tales que $I(p) = \emptyset$.

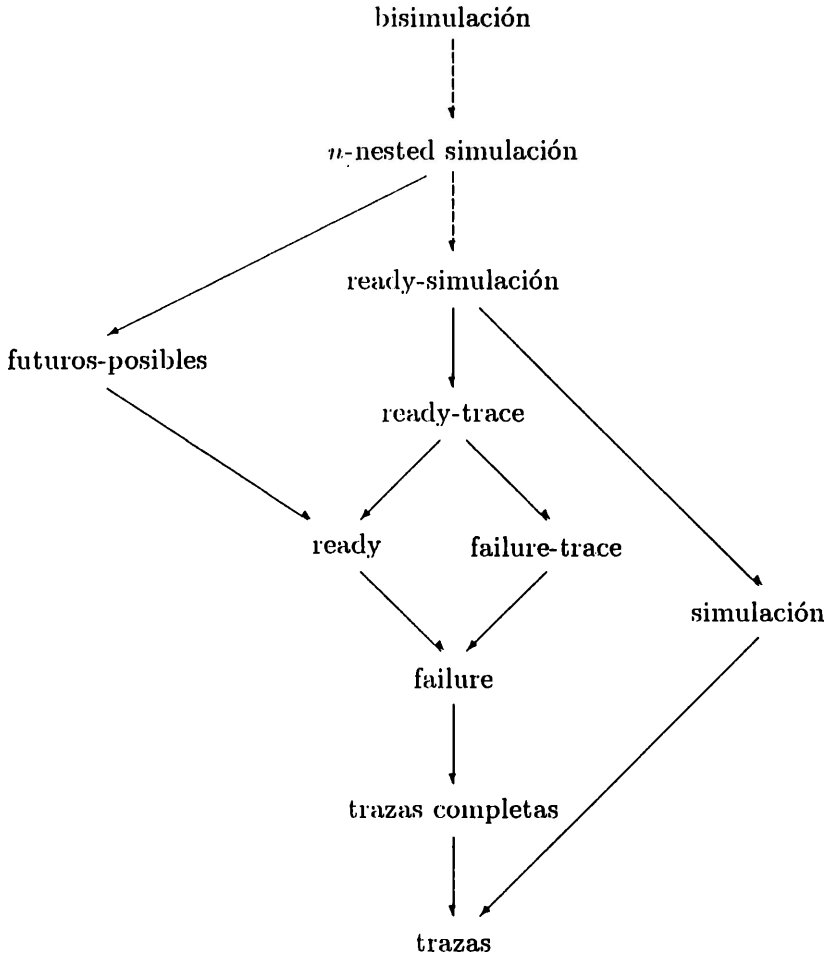


Figura 2.3: Reticulado de equivalencias semánticas

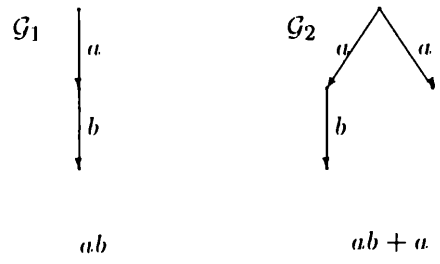


Figura 2.4: STE equivalentes por trazas

Definición 2.3.2 (Trazas completas) *Una traza* $\sigma \in Act^*$ *del STE* $\mathcal{G} = (S, i, Act, \longrightarrow)$ *es* completa *si existe* $p \in S$ *tal que* $i \xrightarrow{\sigma} p$ *y* $I(p) = \emptyset$. *Llamamos* $TC(\mathcal{G})$ *al conjunto de trazas completas de* $\mathcal{G}$. *Decimos que dos STE,* $\mathcal{G}_1, \mathcal{G}_2$ *son* equivalentes por trazas completas, $(\mathcal{G}_1 \equiv_{TC} \mathcal{G}_2)$ *sii* $TC(\mathcal{G}_1) = TC(\mathcal{G}_2)$.

Los STE de la figura 2.4 no son equivalentes por trazas completas ya que $a \in TC(\mathcal{G}_2)$ y $a \notin TC(\mathcal{G}_1)$. Si, en cambio, lo son los STE de la figura 2.5 ($TC(\mathcal{G}_1) = \{ab, ac\} = TC(\mathcal{G}_2)$).

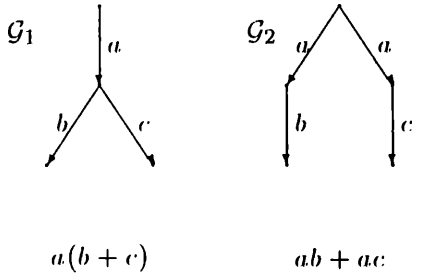


Figura 2.5: STE equivalentes por trazas completas

Más fina es la *equivalencia por falla* o *failure equivalencia* [BHR84, Hoa85]. Para definirla se usan los *pares failure* formados por una traza y un conjunto de acciones, llamado *conjunto failure*. Intuitivamente un conjunto de falla contiene las acciones que un estado no puede realizar.

Definición 2.3.3 (Failure) *El par* $< \sigma, X > \in Act^* \times P(Act)$ *es un* par failure *de un STE* $\mathcal{G} = (S, i, Act, \longrightarrow)$ *si existe* $p \in S$ *tal que* $i \xrightarrow{\sigma} p$ *y* $I(p) \cap X = \emptyset$. *Sea* $F(\mathcal{G})$ *el conjunto de pares failure de* $\mathcal{G}$. *Dos STE* $\mathcal{G}_1, \mathcal{G}_2$ *son failure equivalentes* $(\mathcal{G}_1 \equiv_F \mathcal{G}_2)$ *sii* $F(\mathcal{G}_1) = F(\mathcal{G}_2)$.

Vemos que los STE $\mathcal{G}_1$ y $\mathcal{G}_2$ de la figura 2.5 no son equivalentes por failure porque $< a, \{b\} > \in F(\mathcal{G}_2) \wedge < a, \{b\} > \notin F(\mathcal{G}_1)$.

Para refinar aún mas las equivalencias podemos incorporar la idea de observaciones completas a la idea de failure, obteniendo las *failure-trace* o *trazas de falla* [Phi87], formadas por secuencias de pares traza-conjunto failure.

Para definirla usamos $p \xrightarrow{X} q$, con $X \subseteq Act$ sii $p = q \wedge I(p) \cap X = \emptyset$. También definimos $p \xrightarrow{<\sigma, X>} q$ sii $p \xrightarrow{\sigma} r \xrightarrow{X} q$. El significado de la relación $\longrightarrow$ se deduce claramente del contexto.

Más aún, usamos $p \xrightarrow{\omega} q$ para denotar $p_1 \xrightarrow{<\sigma_1,X_1>} p_2 \ldots p_{n-1} \xrightarrow{<\sigma_m,X_m>} p_n$, con $\omega =< \sigma_1, X_1 > \ldots < \sigma_m, X_m >$.

Definición 2.3.4 (Failure-trace) *$\omega \in (Act^* \times P(Act))^*$ es una traza de failure de un STE $\mathcal{G} = (S, i, Act, \longrightarrow)$ si existe $q \in S$ tal que $i \xrightarrow{\omega} q$ y $I(q) = \emptyset$. Sea $FT(\mathcal{G})$ el conjunto de failure-traces de $\mathcal{G}$. Dos STE $\mathcal{G}_1, \mathcal{G}_2$ son failure-trace equivalentes ($\mathcal{G}_1 \equiv_{FT} \mathcal{G}_2$) sii $FT(\mathcal{G}_1) = FT(\mathcal{G}_2)$.*

En la figura 2.6 vemos un ejemplo de STE failure-trace equivalentes. Allí se puede observar la failure-trace $< a, \{c\} >< b, \{a, b\} >$.

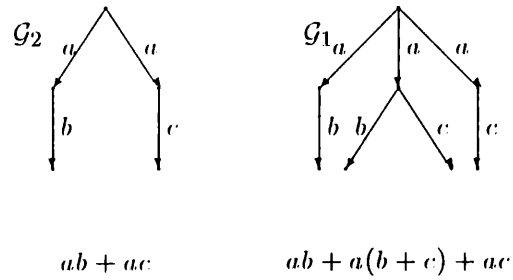


Figura 2.6: STE failure-trace equivalentes

Note que la secuencia de pares $< \sigma, \{a, b\} >< a, X >$ no puede darse nunca como failure-trace de un STE. El conjunto failure $\{a, b\}$ implica que a no es una acción que pueda realizar el STE después de realizar la secuencia σ.

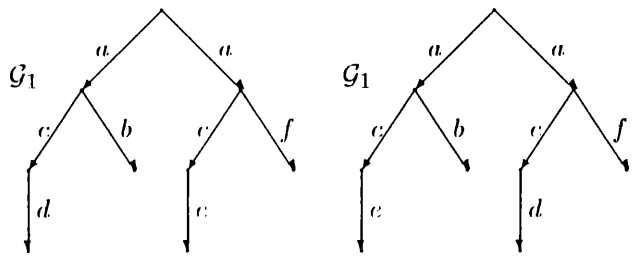
Una equivalencia más fina que la anterior es la *ready equivalencia* [OH83]. Es en algún sentido simétrica a la failure-trace, considerando pares formados por trazas y conjuntos de acciones que *sí* pueden realizarse.

Definición 2.3.5 (Ready) *El par $< \sigma, X >\in Act^* \times P(Act)$ es un* par ready *de un STE $\mathcal{G} = (S, i, Act, \longrightarrow)$ si existe $p \in S$ tal que $i \xrightarrow{\sigma} p$ y $I(p) = X$. Sea $Re(\mathcal{G})$ el conjunto de pares ready de $\mathcal{G}$. Dos STE $\mathcal{G}_1, \mathcal{G}_2$ son ready equivalentes ($\mathcal{G}_1 \equiv_{Re} \mathcal{G}_2$) sii $Re(\mathcal{G}_1) = Re(\mathcal{G}_2)$.*

La ready equivalencia es incomparable con la failure-trace equivalencia, ésto es, dos STE ready equivalentes no necesariamente son failure-trace equivalentes y viceversa (ver el reticulado de semánticas en la figura 2.3). En la figura 2.6 podemos ver, con el par ready $< a, \{b, c\} >$, que los STE failure-trace equivalentes no son ready equivalentes: $< a, \{b, c\} > \in Re(\mathcal{G}_2) \wedge < a, \{b, c\} > \notin Re(\mathcal{G}_1)$. Un ejemplo de STE ready equivalentes se puede ver en la figura 2.7. Los mismos no son failure-trace equivalentes porque: $< a, \{f\} >< c, \{e\} > < d, \emptyset >\in FT(\mathcal{G}_1)$ y $< a, \{f\} >< c, \{e\} >< d, \emptyset > \notin FT(\mathcal{G}_2)$.

De la misma manera en que obtuvimos la failure-trace equivalencia a partir de la failure equivalencia podemos definir una equivalencia ready-trace [Pnu85, BBK85]. Esta equivalencia, mas fina que la ready equivalencia y que la failure-trace equivalencia, se define a partir de secuencias de pares traza-conjunto de inicios.

Para definir la ready-trace equivalencia definimos $p \xrightarrow{X} q$, con $X \subseteq Act$ sii $p = q \wedge I(p) = X$, además de $p \xrightarrow{<\sigma,X>} q$ sii $p \xrightarrow{\sigma} r \xrightarrow{X} q$. Esto recarga aún mas el uso de la relación $\longrightarrow$ pero creemos que el significado sigue siendo deducido fácilmente del contexto.

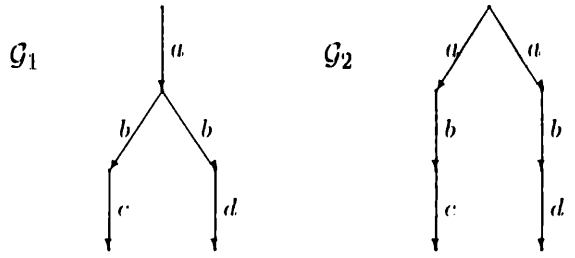


$a(cd + b) + a(ce + f)$ $\quad$ $a(ce + b) + a(cd + f)$

Figura 2.7: STE ready equivalentes

Definición 2.3.6 (Ready-trace) *$\omega \in (Act^* \times P(Act))^*$ es una ready-trace de un STE $\mathcal{G} = (S, i, Act, \longrightarrow)$ si existe $q \in S$ tal que $i \xrightarrow{\omega} q$ y $I(q) = \emptyset$. Sea $RT(\mathcal{G})$ el conjunto de ready-traces de $\mathcal{G}$. Dos STE $\mathcal{G}_1, \mathcal{G}_2$ son ready-trace equivalentes ($\mathcal{G}_1 \equiv_{RT} \mathcal{G}_2$) sii $RT(\mathcal{G}_1) = RT(\mathcal{G}_2)$.*

Los STE de la figura 2.7 no son ready-trace equivalentes, basta tomar $\omega = < a, \{b, c\} > < c, \{d\} >$ y ver que $\omega \in RT(\mathcal{G}_1)$ y $\omega \notin RT(\mathcal{G}_2)$. Dos STE ready-trace equivalentes se pueden ver en la figura 2.8.



$a(bc + bd)$ $\quad$ $abc + abd$

Figura 2.8: STE ready-trace equivalentes

Una semántica mas fina aún, no muy frecuente en la literatura, se obtiene a partir de la equivalencia de futuros-posibles [RS81]. Los futuros posibles de un STE están dados por un conjunto de pares traza-conjunto de trazas, en donde el conjunto contiene todas las continuaciones que se pueden originar a partir de la traza original.

Definición 2.3.7 (Futuros-posibles) *Sea $\mathcal{G} = (S, i, Act, \longrightarrow)$ un STE y $\sigma \in Act^*$. Definimos $FP(\mathcal{G})$ como el conjunto de pares $< \sigma, T >$, tal que $\exists p \in S$ tal que $i \xrightarrow{\sigma} p$ y $T = TC((S, p, Act, \longrightarrow))$. Dos STE, $\mathcal{G}_1, \mathcal{G}_2$ son* equivalentes por futuros-posibles *($\mathcal{G}_1 \equiv_{FP} \mathcal{G}_2$) sii $FP(\mathcal{G}_1) = FP(\mathcal{G}_2)$.*

Sea $\mathcal{G} = (S, i, Act, \longrightarrow)$ *un STE y* $\sigma \in Act^*$ *una traza tal que* $i \xrightarrow{\sigma} p$. *Definimos el conjunto* $TF(p) = \{\phi : p \xrightarrow{\phi} q \wedge I(q) = \emptyset\}$ *y llamamos* $FP(\mathcal{G})$ *al conjunto de pares* $< \sigma, TF(p) >$. *Dos STE,* $\mathcal{G}_1, \mathcal{G}_2$ *son* equivalentes por futuros-posibles *(*$\mathcal{G}_1 \equiv_{FP} \mathcal{G}_2$*) sii* $FP(\mathcal{G}_1) = FP(\mathcal{G}_2)$.

Los STE $\mathcal{G}_1$ y $\mathcal{G}_2$ de la figura 2.9 son equivalentes por futuros-posibles. No así los $\mathcal{G}_1$ y $\mathcal{G}_2$ de las figuras 2.8 y 2.7, que demostramos con los pares $< a, \{bc, bd\} >$ y $< a, \{cd, b\} >$.

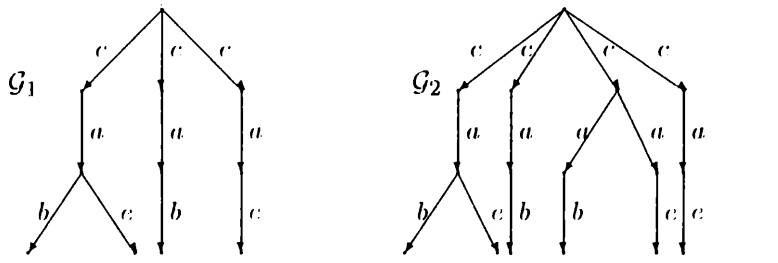


Figura 2.9: STE equivalentes por futuros-posibles

Las equivalencias restantes se definen a partir de distintas variantes de un juego de imitación. Dos STE serán equivalentes cuando puedan resolver exitosamente el juego de imitación propuesto.

Intuitivamente un STE *simula* a otro cuando es posible encontrar una estrategia que permita a un STE imitar la evolución del otro: cuando este último ejecuta (o "propone") una acción el primero lo imita realizando la misma acción, llegando a un estado que simula el estado al que llegó el anterior. Esta estrategia, llamada *simulación* [Par81], consiste en un conjunto de pares de estado (p, q) donde q puede imitar (en un sentido muy preciso) las evoluciones del estado p. Dos STE serán similares cuando se puedan encontrar estrategias para que cada uno simule al otro.

Al pedir que dos procesos se identifiquen cuando son similares obtenemos una semántica más fina que la de trazas, aunque incomparable con las demás vistas hasta ahora (ver la figura 2.3).

Definición 2.3.8 (Simulación) *Sean* $\mathcal{G}_1 = (S_1, i_1, Act, \longrightarrow_1), \mathcal{G}_2 = (S_2, i_2, Act, \longrightarrow_2) \in \mathcal{C}_{STE}$. *Una* simulación $R \subseteq S_1 \times S_2$ *de* $\mathcal{G}_1$ *en* $\mathcal{G}_2$ *(*$R : \mathcal{G}_1 \subseteq\!\!\!\!\supseteq \mathcal{G}_2$*) es una relación binaria que satisface* $i_1 R i_2$ *y, para* $p \in S_1, q \in S_2$,

si pRq *y* $p \xrightarrow{a}_1 p'$, *entonces* $\exists q' : q \xrightarrow{a}_2 q'$ *y* $p' R q'$

Un STE $\mathcal{G}_1$ *es simulado por* $\mathcal{G}_2$ *(*$\mathcal{G}_1 \subseteq\!\!\!\!\supseteq \mathcal{G}_2$*) si existe* $R : \mathcal{G}_1 \subseteq\!\!\!\!\supseteq \mathcal{G}_2$. *Decimos que* $\mathcal{G}_1$ *y* $\mathcal{G}_2$ *son* similares *(*$\mathcal{G}_1 \rightleftharpoons \mathcal{G}_2$*) sii* $\mathcal{G}_1 \subseteq\!\!\!\!\supseteq \mathcal{G}_2$ *y* $\mathcal{G}_2 \subseteq\!\!\!\!\supseteq \mathcal{G}_1$.

Esto es, dos STE son similares cuando es posible encontrar una estrategia para que cada STE imite al otro. Estas dos estrategias no son necesariamente la misma (esto es una la

inversa de la otra), el par (p, q) en una de ellas significa que q puede imitar a p pero no necesariamente p puede imitar a q: los roles de imitado e imitador no son intercambiables.

Los STE $\mathcal{G}_1$ y $\mathcal{G}_2$ de la figura 2.8 no son similares. $\mathcal{G}_1$ puede imitar al STE $\mathcal{G}_2$ pero $\mathcal{G}_2$ no puede imitar a $\mathcal{G}_1$. Cuando $\mathcal{G}_1$ realiza la acción a no se sabe si luego va a realizar bc o bd. Así $\mathcal{G}_2$ no sabe como imitarla. No importa cual es la acción a que $\mathcal{G}_2$ realiza, siempre $\mathcal{G}_1$ va a poder proponer una acción que no va a poder ser imitada.

En la figura 2.10 vemos dos STE $\mathcal{G}_1$ y $\mathcal{G}_2$ que son similares. Usando la relación de identidad entre los estados $Id \subseteq S_1 \times S_2$ podemos escribir ambas estrategias de simulación: $R_1 = Id \cup \{(a(bd + cd), a(bd + cd) + acd)\}$ y $R_2 = Id \cup \{(a(bd + cd), a(bd + cd) + acd), (bd + cd, cd)\}$. $R_1 : \mathcal{G}_1 \subseteq_{\rightarrow} \mathcal{G}_2$ y $R_2 : \mathcal{G}_2 \subseteq_{\rightarrow} \mathcal{G}_1$.

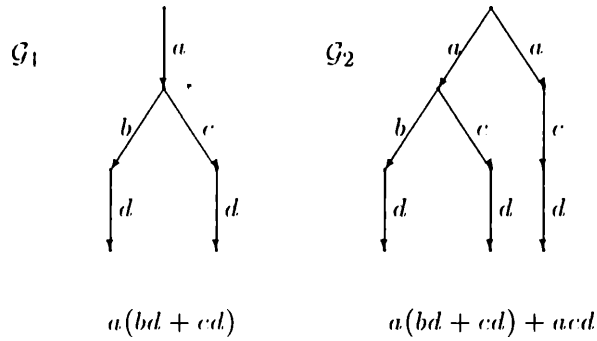


Figura 2.10: STE similares

Otra equivalencia que aparece con frecuencia en la literatura es la *ready-simulación* [BIM88]. Consiste en un refinamiento de la simulación, en donde se pide la preservación de las acciones iniciales para los estados que se simulan.

Definición 2.3.9 (Ready-simulación) *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$. Una* ready-simulación $R \subseteq S_1 \times S_2$ *de $\mathcal{G}_1$ en $\mathcal{G}_2$, notación $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$, es una simulación $R : \mathcal{G}_1 \subseteq_{\rightarrow} \mathcal{G}_2$ que satisface $\forall p \in S_1, q \in S_2$: si pRq entonces $I(p) = I(q)$.*

Un STE $\mathcal{G}_1$ es ready-simulado por $\mathcal{G}_2$, notación $\mathcal{G}_1 \subseteq_R \mathcal{G}_2$, si existe $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$. Decimos que $\mathcal{G}_1$ y $\mathcal{G}_2$ son ready-similares, *notación $\mathcal{G}_1 \rightleftharpoons_R \mathcal{G}_2$ sii $\mathcal{G}_1 \subseteq_R \mathcal{G}_2$ y $\mathcal{G}_2 \subseteq_R \mathcal{G}_1$.*

La ready-simulación exige que el conjunto de acciones iniciales sea el mismo en los dos STE simulados y, del mismo modo que en simulación, no se permite el intercambio de roles. La ready-simulación se encuentra entre la simulación (los STE $\mathcal{G}_1$ y $\mathcal{G}_2$ de la figura 2.10 son similares pero no ready-similares) y las n-nested simulaciones.

Volvamos al juego original de simulación. Si modificamos sus reglas de manera de permitir el intercambio de roles en cualquier momento de la imitación obtenemos una equivalencia mas fina que la anterior, la *bisimulación* [Par81]. Este intercambio, de imitado e imitador, puede realizarse tantas veces como se desee a lo largo del juego: un STE propone una acción y a continuación, después de imitar la acción, el otro STE es el que propone. Para obtener una bisimulación alcanza con pedir que una simulación sea la inversa de la otra, en el sentido de

que cada par (p, q) significa q simula p y p simula a q. La bisimulación es la equivalencia más fina que vamos a analizar en este trabajo.

Definición 2.3.10 (Bisimulación) *Sean* $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$. *Una* bisimulación $R \subseteq S_1 \times S_2$ *entre* $\mathcal{G}_1$ *y* $\mathcal{G}_2$, *notación* $R : \mathcal{G}_1 \leftrightarrows \mathcal{G}_2$ *es una relación binaria que satisface* $R : \mathcal{G}_1 \subseteq_{\rightarrow} \mathcal{G}_2$ *y* $R^{-1} : \mathcal{G}_2 \subseteq_{\rightarrow} \mathcal{G}_1$.

$\mathcal{G}_1$ *y* $\mathcal{G}_2$ *son* bisimilares, *notación* $\mathcal{G}_1 \leftrightarrows \mathcal{G}_2$, *sii existe* $R : \mathcal{G}_1 \leftrightarrows \mathcal{G}_2$.

En la figura 2.3 se ve que la bisimulación es una equivalencia estrictamente mas fina que la simulación. Efectivamente, en la figura 2.10, vemos dos STE similares que no son bisimilares. En la figura 2.11, en cambio, observamos dos STE bisimilares. No es difícil obtener la relación R que lo demuestra: $R = Id \cup \{(aab, a(ab + ab)), (ab, ab + ab)\}$.

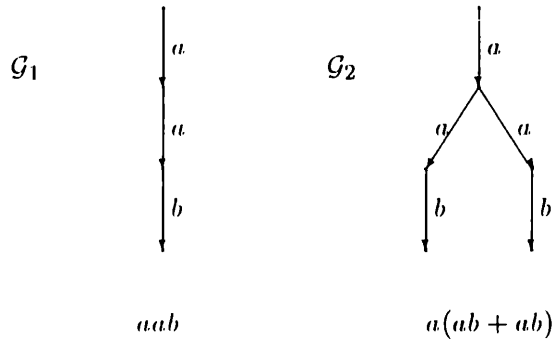


Figura 2.11: STE bisimilares

Las equivalencias que más nos van a interesar en este trabajo conforman una opción intermedia a las vistas. Se las conoce como n-nested simulaciones, que forman una familia infinita ubicada entre la simulación y la bisimulación. A lo largo del juego de imitación se permite un número fijo de cambios de roles. Para cada n, surge la noción de *n-nested simulación* [GV89] en donde los roles de imitado e imitador pueden ser cambiados a lo sumo $n - 1$ veces.

Definición 2.3.11 (n-nested simulación) *Sean* $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$. *Una* n-nested simulación $R \subseteq S_1 \times S_2$ *de* $\mathcal{G}_1$ *en* $\mathcal{G}_2$, *notación* $R : \mathcal{G}_1 \subseteq_{n} \mathcal{G}_2$, *es una relación que satisaface:*

$R : \mathcal{G}_1 \subseteq_{0} \mathcal{G}_2$ *sii* $i_1 R i_2$

$R : \mathcal{G}_1 \subseteq_{n+1} \mathcal{G}_2$ *sii* $R : \mathcal{G}_1 \subseteq \mathcal{G}_2 \wedge \exists Q : \mathcal{G}_2 \subseteq_{n} \mathcal{G}_1$ *con* $R^{-1} \subseteq Q$

$\mathcal{G}_1$ *es* n-nested simulado por $\mathcal{G}_2$, *notación* $\mathcal{G}_1 \subseteq_{n} \mathcal{G}_2$, *si existe* $R : \mathcal{G}_1 \subseteq_{n} \mathcal{G}_2$. $\mathcal{G}_1$ *y* $\mathcal{G}_2$ *son* n-nested similares, *notación* $\mathcal{G}_1 \rightleftharpoons_n \mathcal{G}_2$, *sii* $\mathcal{G}_1 \subseteq_{n} \mathcal{G}_2$ *y* $\mathcal{G}_2 \subseteq_{n} \mathcal{G}_1$.

Nota: La 1-nested simulación ($\rightleftharpoons_1$) coincide con la simulación ($\rightleftharpoons$). ■

Ejemplo 2.3.12 ([GV89]) *Veamos que los STE de la figura 2.12 son 2-nested similares. Para eso damos relaciones R_2, R_1 y Q_2, Q_1 tales que $R_2 : \mathcal{G}_1 \subseteq_2 \mathcal{G}_2$ y $Q_2 : \mathcal{G}_2 \subseteq_2 \mathcal{G}_1$.*

Siendo Id la identidad entre estados, construimos:

$$R_2 = Id \cup \{(a(a(b+c)+ab), a(a(b+c)+ab)+aa(b+c))\}$$

$$R_1 = R_2^{-1} \cup \{(a(b+c), a(b+c)+ab)\}$$

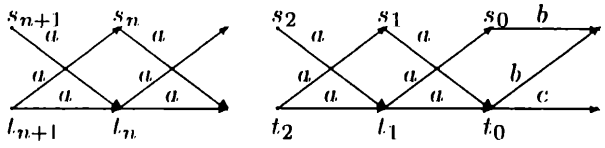
$$Q_2 = R_1$$

$$Q_1 = Q_2^{-1} \cup \{(b, b+c)\}$$

Se puede ver fácilmente que éstas relaciones cumplen con las condiciones de la definición. En la próxima sección vemos que éstos $\mathcal{G}_1$ y $\mathcal{G}_2$ no son 3-nested similares.

Las n-nested simulaciones forman una familia infinita de equivalencias. Para cada natural n obtenemos una equivalencia diferente, mas fina cuando n crece, y todas ellas estrictamente mas gruesas que la bisimulación (los STE de la figura 2.12 no son bisimilares). Los STE del ejemplo 2.3.12 resultan sumamente interesantes ya que a partir de ellos se puede obtener, para cada n, un par de STE n-nested similares:

Ejemplo 2.3.13 *En* [GV89] *se generalizan los STE $\mathcal{G}_1$ y $\mathcal{G}_2$ del ejemplo 2.3.12 para obtener $\mathcal{G}_1^n$ y $\mathcal{G}_2^n$, tales que $\mathcal{G}_1^n \rightleftharpoons_n \mathcal{G}_2^n$ y $\mathcal{G}_1^n \not\rightleftharpoons_{n+1} \mathcal{G}_2^n$, para $n \geq 2$. Esos STE se obtienen a partir de los términos s_n y t_n de la siguiente figura*



los cuales se definen, para $n \geq 0$ por:

$$s_0 = b \qquad t_0 = b + c$$
$$s_{n+1} = a.t_n \qquad t_{n+1} = a.s_n + a.t_n$$

Más adelante se demuestra que $\mathcal{G}_1^n \not\rightleftharpoons_{n+1} \mathcal{G}_2^n$.

Nota: Para $n \geq 2$, $\leftrightarrows \subset \rightleftharpoons_{n+1} \subset \rightleftharpoons_n \subset \rightleftharpoons_R \subset \rightleftharpoons$. ■

2.4 Lógica HML

En el marco de una lógica, la semántica de un proceso se define como el conjunto de fórmulas de la lógica que éste satisface. En este trabajo nos interesamos por las equivalencias semánticas de la familia de la simulación. Por eso es que definimos la lógica de Hennessy y Milner que nos permite caracterizarlas. Es en éste marco es que se realizan la mayoría de las demostraciones sobre los procesos.

Definición 2.4.1 (Lógica HML [HM85]) *Dado un alfabeto* $\{a, b, \ldots\}$, *definimos el conjunto* $\mathcal{L}_{HML}$ *de* fórmulas de la lógica de Hennessy y Milner (HML) *mediante la siguiente gramática:*

$$\phi ::= T \mid \phi \wedge \phi \mid \neg\phi \mid \langle a\rangle\phi$$

Damos ahora una relación que indica cuando un proceso o estado satisface una fórmula. La misma responde al significado intuitivo del lenguaje:

Definición 2.4.2 (Relación de satisfacción) *Sea* $p \in S$ *un estado de un proceso. La* relación de satisfacción $\models \subseteq S \times \mathcal{L}_{HML}$ *se define de la siguiente manera:*

- $p \models T$, *para todo* p.
- $p \models \varphi \wedge \psi$ *sii* $p \models \varphi$ *y* $p \models \psi$.
- $p \models \neg\varphi$ *sii* $p \not\models \varphi$.
- $p \models \langle a\rangle\varphi$ *sii existe* $q \in S$ *tal que* $p \xrightarrow{a} q$ *y* $q \models \varphi$.

Dado un conjunto de fórmulas definimos ahora la idea de "equivalencia inducida". Si tenemos un conjunto χ de fórmulas decimos que dos procesos son equivalentes bajo la equivalencia inducida por χ cuando satisfacen exactamente las mismas fórmulas del conjunto. Nosotros vamos a tomar subconjuntos del conjunto de fórmulas HML y vamos a considerar las equivalencias inducidas por ellos.

Definición 2.4.3 (Relación de equivalencia inducida por una lógica) *Sean* $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$ *y sea* $\chi \subseteq \mathcal{L}_{HML}$ *un conjunto de fórmulas HML. Con* $\sim_\chi$ *denotamos la relación de equivalencia en* $\mathcal{C}_{STE}$ *inducida por el conjunto* χ*:*

$$\mathcal{G}_1 \sim_\chi \mathcal{G}_2 \stackrel{\text{def}}{\Leftrightarrow} (\forall \phi \in \chi : i_1 \models \phi \Leftrightarrow i_2 \models \phi)$$

En dónde i_1 *e* i_2 *corresponden a los estados iniciales de* $\mathcal{G}_1$ *y* $\mathcal{G}_2$. *Llamamos* equivalencia χ-fórmula *a ésta relación.*

Nota: En general abusamos de la notación y usamos $\mathcal{G} \models \phi$ significando $i \models \phi$, en donde i corresponde al estado inicial de $\mathcal{G}$. ■

Si tomamos, por ejemplo, el conjunto completo $\mathcal{L}_{HML}$ y consideramos la equivalencia inducida por él obtenemos exactamente la bisimulación [HM85]. Esta caracterización fue la que dió origen a todas las demás.

Teorema 2.4.4 (Bisimulación en HML) *Sean* $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$, *de imagen finita.* $\mathcal{G}_1 \leftrightarrows \mathcal{G}_2 \Leftrightarrow \mathcal{G}_1 \sim_{\mathcal{L}_{HML}} \mathcal{G}_2$

Para caracterizar la familia de las n-nested simulaciones se restringe el conjunto de fórmulas HML, limitando la anidación de los símbolos $\neg$ [GV89].

Definición 2.4.5 *Para todo* $n \in \mathbf{N}$ *definimos el conjunto* $\mathcal{L}_n$ *de fórmulas HML de la siguiente manera:*

- $\mathcal{L}_0$ *es vacío*
- $\mathcal{L}_{n+1}$ *está dado por la gramática* $\phi ::= T \mid \phi \wedge \phi \mid \langle a \rangle \phi \mid \neg \psi$ *(para* $\psi \in \mathcal{L}_n$*)*

Si uno piensa detenidamente en el significado de los símbolos de negación en las fórmulas se da cuenta de que sus ocurrencias están fuertemente relacionadas con los cambios de roles que se realizan en el juego de imitación. Note que el conjunto de fórmulas $\mathcal{L}_n$ coincide con las fórmulas de HML con a lo sumo $n-1$ símbolos $\neg$ anidados. Esto es, se permiten a lo sumo $n-1$ cambios de roles.
Nota: Para cada n, $\mathcal{L}_n \subseteq \mathcal{L}_{n+1}$. ■

De esta manera, restringiendo las fórmulas de HML a las $\mathcal{L}n$, obtenemos un resultado similar al teorema 2.4.4 para las n-nested simulaciones en HML [GV89]:

Teorema 2.4.6 (n-nested simulación en HML) *Sean* $\mathcal{G}_1$ *y* $\mathcal{G}_2$ *STE de imagen finita y* $n \in \mathbf{N}$. $\mathcal{G}_1 \rightleftharpoons_n \mathcal{G}_2 \Leftrightarrow \mathcal{G}_1 \sim_{\mathcal{L}_n} \mathcal{G}_2$

Inmediatamente de la definición de n-nested simulación y el teorema anterior obtenemos la caracterización para la simulación.

Corolario 2.4.7 (Simulación en HML) *Sean* $\mathcal{G}_1$ *y* $\mathcal{G}_2$ *STE de imagen finita.* $\mathcal{G}_1 \rightleftharpoons \mathcal{G}_2 \Leftrightarrow \mathcal{G}_1 \sim_{\mathcal{L}_1} \mathcal{G}_2$

Para dar la caraterización de la ready-simulación es necesario extender el conjunto $\mathcal{L}_{HML}$ de fórmulas HML [BIM88, Gla90]. Necesitamos una manera de verificar que los conjuntos de inicios sean iguales.

Definición 2.4.8 (Extensión de HML para ready-simulación) *Dado un alfabeto de acciones Act y* $A \in Act$*, definimos el conjunto* $\mathcal{L}_R$ *de* fórmulas HML *mediante la siguiente gramática:*

$$\phi ::= T \mid \phi \wedge \phi \mid A \mid \langle a \rangle \phi$$

Para la nueva fórmula A damos la relación de satisfacción: $p \models A$ *sii* $I(p) = A$.

Ahora sí podemos dar la caracterización de la ready-simulación a partir del conjunto $\mathcal{L}_R$ de fórmulas HML.

Teorema 2.4.9 (Ready-simulación en HML) *Sean los STE* $\mathcal{G}_1$ *y* $\mathcal{G}_2$ *de imagen finita.* $\mathcal{G}_1 \rightleftharpoons_R \mathcal{G}_2 \Leftrightarrow \mathcal{G}_1 \sim_{\mathcal{L}_R} \mathcal{G}_2$

A continuación vamos a mostrar como se realizan las pruebas en este formalismo. Veamos un par de demostraciones de procesos que satisfacen o no una serie de fórmulas.

Lema 2.4.10 *Sea* $f \in \mathcal{L}_1$*, tal que* $f = \langle a \rangle(\langle b \rangle \langle c \rangle T \wedge \langle b \rangle \langle d \rangle T)$ *y sean* $\mathcal{G}_1$ *y* $\mathcal{G}_2$ *los STE de la figura 2.8. Entonces* $\mathcal{G}_1 \models f$ *y* $\mathcal{G}_2 \not\models f$.

Demostración: En el caso del primer proceso: $\mathcal{G}_1 \models f \Leftrightarrow a(bc+bd) \models \langle a\rangle(\langle b\rangle\langle c\rangle T \wedge \langle b\rangle\langle d\rangle T) \Leftrightarrow bc+bd \models \langle b\rangle\langle c\rangle T \wedge \langle b\rangle\langle d\rangle T \Leftarrow bc \models \langle b\rangle\langle c\rangle T \wedge bd \models \langle b\rangle\langle d\rangle Tc \models \langle c\rangle T \wedge d \models \langle b\rangle T$.

Además $\mathcal{G}_2 \not\models f \Leftrightarrow abc+abd \not\models \langle a\rangle(\langle b\rangle\langle c\rangle T \wedge \langle b\rangle\langle d\rangle T) \Leftrightarrow bc \not\models (\langle b\rangle\langle c\rangle T \wedge \langle b\rangle\langle d\rangle T) \wedge bd \not\models (\langle b\rangle\langle c\rangle T \wedge \langle b\rangle\langle d\rangle T) \Leftrightarrow (bc \not\models \langle b\rangle\langle c\rangle T \emptyset bc \not\models \langle b\rangle\langle d\rangle T) \wedge (bd \not\models \langle b\rangle\langle c\rangle T \emptyset bd \not\models \langle b\rangle\langle d\rangle T) \Leftarrow bc \not\models \langle b\rangle\langle d\rangle T \wedge bd \not\models \langle b\rangle\langle c\rangle T \Leftrightarrow c \not\models \langle d\rangle T \wedge d \not\models \langle c\rangle T$. ■

Cuando consideramos el conjunto extendido de fórmulas $\mathcal{L}_R$ encontramos algunas diferencias en las demostraciones, aunque no se presenta ninguna dificultad. Veamos un ejemplo:

Lema 2.4.11 *Sea $f \in \mathcal{L}_R, f = \langle a\rangle\{c\}$ y sean $\mathcal{G}_1$ y $\mathcal{G}_2$ los STE n-nested similares del ejemplo 2.10. Entonces $\mathcal{G}_2 \models f$ y $\mathcal{G}_1 \not\models f$.*

Demostración: $\mathcal{G}_1 \not\models f \Leftrightarrow a(bd+cd) \not\models \langle a\rangle\{c\} \Leftrightarrow bd+cd \not\models \{c\}$. El proceso $bd+cd$ no satisface al conjunto $\{c\}$ porque $I(bd+cd) = \{b,c\} \neq \{c\}$.

$\mathcal{G}_2 \models f \Leftrightarrow a(bd+cd)+acd \models \langle a\rangle\{c\} \Leftrightarrow a(bd+cd) \models \langle a\rangle\{c\} \vee acd \models \langle a\rangle\{c\} \Leftarrow acd \models \langle a\rangle\{c\} \Leftrightarrow cd \models \{c\}$. ■

Veamos ahora como se demuestra en este marco que dos procesos no son equivalentes. Por ejemplo demostremos que dos procesos no son bisimilares. En primer lugar buscamos una fórmula del conjunto $\mathcal{L}_{HML}$ tal que sólo la satisfaga uno de los procesos.

Lema 2.4.12 *Sean $\mathcal{G}_1$ y $\mathcal{G}_2$ los STE del ejemplo 2.3.12 y sea $f \in \mathcal{L}_{HML} : f = \langle a\rangle\neg\langle a\rangle\neg\langle c\rangle T$. Entonces $\mathcal{G}_2 \models f$ y $\mathcal{G}_1 \not\models f$.*

Demostración: $\mathcal{G}_1 \not\models f \Leftrightarrow a(a(b+c)+ab) \not\models \langle a\rangle\neg\langle a\rangle\neg\langle c\rangle T \Leftrightarrow a(b+c)+ab \not\models \neg\langle a\rangle\neg\langle c\rangle T \Leftrightarrow a(b+c)+ab \models \langle a\rangle\neg\langle c\rangle T \Leftrightarrow a(b+c) \models \langle a\rangle\neg\langle c\rangle T \vee ab \models \langle a\rangle\neg\langle c\rangle T \Leftarrow ab \models \langle a\rangle\neg\langle c\rangle T \Leftrightarrow b \models \neg\langle c\rangle T \Leftrightarrow b \not\models \langle c\rangle T$.

$\mathcal{G}_2 \models f \Leftrightarrow a(a(b+c)+ab)+aa(b+c) \models \langle a\rangle\neg\langle a\rangle\neg\langle c\rangle T \Leftrightarrow a(a(b+c)+ab) \models \langle a\rangle\neg\langle a\rangle\neg\langle c\rangle T \vee aa(b+c) \models \langle a\rangle\neg\langle a\rangle\neg\langle c\rangle T \Leftarrow a(b+c) \models \neg\langle a\rangle\neg\langle c\rangle T \Leftrightarrow a(b+c) \not\models \langle a\rangle\neg\langle c\rangle T \Leftrightarrow b+c \not\models \neg\langle c\rangle T \Leftrightarrow b+c \models \langle c\rangle T \Leftarrow c \models \langle c\rangle T$. ■

Ejemplo 2.4.13 *Continuemos con los STE del ejemplo 2.3.12. Sabemos que la fórmula $f = \langle a\rangle\neg\langle a\rangle\neg\langle c\rangle T \in \mathcal{L}_{HML}$. A partir del lema anterior usamos el teorema 2.4.4 para demostramos que $\mathcal{G}_1$ y $\mathcal{G}_2$ no son bisimilares. Además usamos el teorema 2.4.6 para ver que los procesos ni siquiera son 3-nested similares porque $f \in \mathcal{L}_3$.*

De la misma manera vamos a demostrar que los STE n-nested equivalentes del ejemplo 2.3.13 no son $n+1$-nested equivalentes. Nuevamente en un primer paso vamos a encontrar una fórmula que sólo es satisfecha por uno de los procesos.

Lema 2.4.14 *Sea $f_{n+1} \in \mathcal{L}n+1$ la fórmula HML obtenida a partir de: $f_1 = \langle c\rangle T$ y $f_{n+1} = \langle a\rangle\neg f_n$ y sean $\mathcal{G}_1^n$, $\mathcal{G}_2^n$ los STE del ejemplo 2.3.13. Entonces $\mathcal{G}_2^n \models f_{n+1}$ y $\mathcal{G}_1^n \not\models f_{n+1}$.*

Demostración: Por inducción sobre n:

(caso $n = 1$) $\mathcal{G}_1^1 \not\models f_2 \Leftrightarrow a(b+c) \not\models \langle a\rangle\neg\langle c\rangle T \Leftrightarrow b+c \not\models \neg\langle c\rangle T \Leftrightarrow b+c \models \langle c\rangle T$.

Además $\mathcal{G}_2^1 \models f_2 \Leftrightarrow ab+a(b+c) \models \langle a\rangle\neg\langle c\rangle T \Leftrightarrow ab \models \langle a\rangle\neg\langle c\rangle T \vee a(b+c) \models \langle a\rangle\neg\langle c\rangle T \Leftarrow b \models \neg\langle c\rangle T \Leftrightarrow b \not\models \langle c\rangle T$.

(**caso** $n > 1$) $\mathcal{G}_1^n \not\models f_{n+1} \Leftrightarrow a\mathcal{G}_2^{n-1} \not\models \langle a\rangle\neg f_n \Leftrightarrow \mathcal{G}_2^{n-1} \not\models \neg f_n \Leftrightarrow \mathcal{G}_2^{n-1} \models f_n$, verdadero por hipótesis inductiva.

Finalmente $\mathcal{G}_2^n \models f_{n+1} \Leftrightarrow a\mathcal{G}_1^{n-1} + a\mathcal{G}_2^{n-1} \models \langle a\rangle\neg f_n \Leftrightarrow a\mathcal{G}_1^{n-1} \models \langle a\rangle\neg f_n \vee a\mathcal{G}_2^{n-1} \models \langle a\rangle\neg f_n \Leftarrow \mathcal{G}_1^{n-1} \models \neg f_n \Leftrightarrow \mathcal{G}_1^{n-1} \not\models f_n$, que es verdadero por hipótesis. ■

Ejemplo 2.4.15 *Tomamos ahora los STE $\mathcal{G}_1^n$ y $\mathcal{G}_2^n$ del ejemplo 2.3.13. A partir del teorema 2.4.6, usando el lema anterior, obtenemos que $\mathcal{G}_1^n \not\simeq_{n+1} \mathcal{G}_2^n$. Como $f_{n+1} \in \mathcal{L}_{HML}$ sabemos además que $\mathcal{G}_1^n \not\simeq \mathcal{G}_2^n$.*

De la misma manera se demuestra que dos STE no son similares o ready similares:

Ejemplo 2.4.16 *Dada la fórmula $f = \langle a\rangle(\langle b\rangle\langle c\rangle T \wedge \langle b\rangle\langle d\rangle T) \in \mathcal{L}_1$ y el lema 2.4.10 se ve que los STE $\mathcal{G}_1$ y $\mathcal{G}_2$ de la figura 2.8 no son similares. Y, finalmente, usamos $f = \langle a\rangle\{c\} \in \mathcal{L}_R$ junto con el lema 2.4.11 para probar que los STE $\mathcal{G}_1$ y $\mathcal{G}_2$ de la figura 2.10 no son ready-similares.*

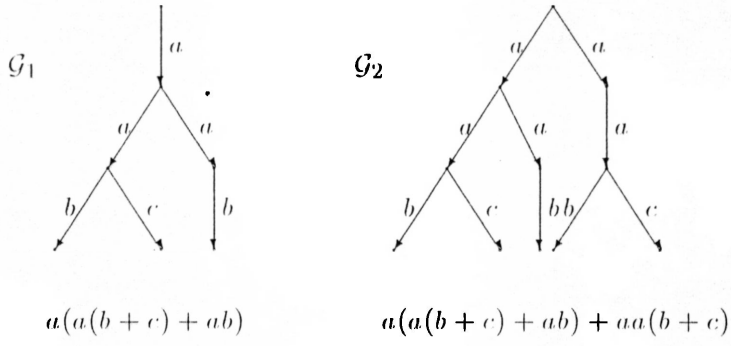


Figura 2.12: STE 2-nested similares

Capítulo 3

Las equivalencias semánticas como congruencias para los operadores de prioridad

En este capítulo se definen formalmente los operadores de prioridad y la noción de congruencia. Se muestra que las únicas equivalencias semánticas vistas que son congruencia para esos operadores son: ready-trace, ready-simulación y bisimulación. Se deja la n-nested simulación para ser analizada en el capítulo siguiente porque presenta numerosas diferencias.

3.1 Operadores de prioridad

Una clase de operadores bastante habitual en la literatura de STE es la clase de los operadores de prioridad $\theta_{\geq}$ [BBK85, BBK86, BW90]. Otros operadores que aparecen frecuentemente y se van a mencionar más adelante son los de renombrado y ocultamiento. En este trabajo se sigue la metodología tradicional de verificación de compatibilidad de un operador con las equivalencias semánticas. Elegimos para ésto la clase de los operadores de prioridad y damos resultados nuevos en cuanto a la compatibilidad de los mismos con respecto a varias de las equivalencias vistas.

Cada operador de prioridad $\theta_{\geq}$ está asociado a un orden parcial $\geq$ en el conjunto de acciones Act. Esta relación, en dónde $a > b$ significa que a tiene prioridad sobre b, cumple $\forall a, b \in Act$:

- $a \geq b$ y $b \geq a$ implica $a = b$,
- $a \geq b$ y $b \geq c$ implica $a \geq c$.

En un STE se le da prioridad a cada transición a partir de la acción que la etiqueta. La aplicación de un operador de prioridad a un proceso consiste en la eliminación de las transiciones de "baja prioridad", dejando sólo las $\geq$-maximales. Esto es, para cada estado del STE se considera cada transición saliente y se conserva solamente si no hay una trancisión mas prioritaria que ella.

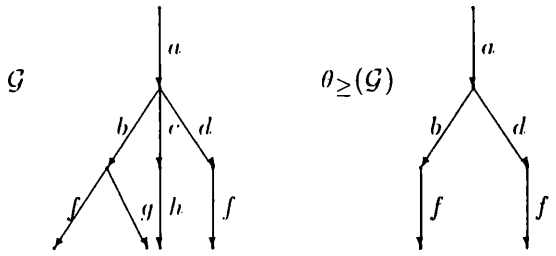
Definición 3.1.1 (Operadores de prioridad) *Dado un STE* $\mathcal{G} = (S, i, Act, \longrightarrow)$ *definimos, para cada orden parcial* $\geq$ *en* Act, *un* operador de prioridad $\theta_{\geq} : C_{STE} \rightarrow C_{STE}$ *tal que:*

$$\theta_{\geq}(S, i, Act, \longrightarrow) = (S, i, Act, \{p \xrightarrow{a}{}^{\theta} q : p \xrightarrow{a} q \wedge \forall b \in I(p), b \not> a\})$$

Cuando no da lugar a confusiones usamos θ en vez de $\theta_{\geq}$.

En el siguiente ejemplo vemos el resultado de la aplicación de un operador de prioridad a un STE.

Ejemplo 3.1.2 *Sea* $\mathcal{G} = (S, i, Act, \longrightarrow)$ *el STE de la figura 3.1. Consideremos en* Act *el orden parcial* $\geq = \{a > d,\ b > c,\ f > g\}$ *y el operador de prioridad obtenido a partir de él. Así obtenemos el* $\theta_{\geq}(\mathcal{G})$ *de la misma figura.*



$a(b(f+g)+ch+df)$ $\qquad$ $a(bf+df)$

Figura 3.1: Aplicación del operador de prioridad

Es importante tener en cuenta que en $\theta_{\geq}(\mathcal{G})$ sólo se eliminan las transiciones y los conjuntos de estados y de acciones se mantienen. De todas formas, por una cuestión de comodidad no dibujamos los estados que quedaron aislados en el STE resultante.

3.2 Congruencia

Si se trabaja con un modelo en particular y se quiere introducir un nuevo operador es natural preguntarse por los resultados de congruencia de las distintas equivalencias semánticas. Esto es, antes de utilizar el operador debemos asegurarnos de que es compatible con las equivalencias elegidas.

Esta situación es similar a cuando se tienen dos implementaciones de una misma función de un sistema. Como responden a una misma especificación resulta lógico querer intercambiarlas. Es de esperar que el sistema no se vea afectado por este cambio, es decir que sea equivalente al inicial.

Definición 3.2.1 (Precongruencia) *Dados* $\mathcal{G}_1, \mathcal{G}_2 \in C_{STE}$, *decimos que un preorden* $\lesssim$ *es una* precongruencia *de* $\mathcal{G}_1$ *en* $\mathcal{G}_2$ *para el operador* $\alpha : C_{STE} \rightarrow C_{STE}$ *sii* $\mathcal{G}_1 \lesssim \mathcal{G}_2 \Rightarrow \alpha(\mathcal{G}_1) \lesssim \alpha(\mathcal{G}_2)$.

Definición 3.2.2 (Congruencia) *Dados $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$, decimos que una equivalencia $\approx$ es una* congruencia *para el operador $\alpha : \mathcal{C}_{STE} \to \mathcal{C}_{STE}$ sii $\mathcal{G}_1 \approx \mathcal{G}_2 \Rightarrow \alpha(\mathcal{G}_1) \approx \alpha(\mathcal{G}_2)$.*

Decimos que una equivalencia $\approx$ es una congruencia para una clase de operadores sii $\approx$ es una congruencia para todos los operadores de la clase.

Nota: Sea $\lesssim$ un preorden tal que es una precongruencia. La equivalencia definida por éste ($\approx = \lesssim \cup \lesssim^{-1}$) es una congruencia. ■

3.3 Equivalencias no preservadas por el operador de prioridad

En esta sección presentamos los resultados negativos sobre la compatibilidad de los operadores de prioridad con respecto a trazas, failure, failure-trace, ready, futuros posibles y simulación, algunos de los cuales ya han aparecido en la literatura (ver [BBK85]). Estos resultados son todos negativos ya que ninguan de estas equivalencias es una congruencia para la clase de operadores de prioridad.

Para demostrar que una equivalencia semántica $\approx$ no es una congruencia para la clase de operadores de prioridad basta con encontrar dos STE equivalentes ($\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$ y $\mathcal{G}_1 \approx \mathcal{G}_2$) y un orden parcial $\geq$ sobre Act tales que $\theta_{\geq}(\mathcal{G}_1) \not\approx \theta_{\geq}(\mathcal{G}_2)$. Una demostración afirmativa, en cambio, es un poco mas laboriosa.

Teorema 3.3.1 *La semántica por trazas no es una congruencia para los operadores de prioridad.*

Demostración: Sean $\mathcal{G}_1$ y $\mathcal{G}_2$ los STE de la figura 3.2. Tenemos que $T(\mathcal{G}_1) = \{a, ab, ac\} = T(\mathcal{G}_2)$, entonces $\mathcal{G}_1 \equiv_T \mathcal{G}_2$. Definimos el orden parcial $\geq = \{b > c\}$ y calculamos $\theta_{\geq}(\mathcal{G}_1)$ y $\theta_{\geq}(\mathcal{G}_2)$ (ver nuevamente la figura 3.2). Observamos que $T(\theta_{\geq}(\mathcal{G}_1)) = \{a, ab\} \neq \{a, ab, ac\} = T(\theta_{\geq}(\mathcal{G}_2))$. Por lo tanto $\theta_{\geq}(\mathcal{G}_1) \not\equiv_T \theta_{\geq}(\mathcal{G}_2))$. ■

Teorema 3.3.2 *La semántica por trazas completas no es una congruencia para los operadores de prioridad.*

Demostración: Ya vimos en la sección anterior que los STE de la figura 3.2 son equivalentes por trazas completas: $TC(\mathcal{G}_1) = \{ab, ac\} = TC(\mathcal{G}_2)$. Pensemos ahora en la demostración del teorema 3.3.1. Allí aplicamos un operador $\theta_{\geq}$ con el que obtuvimos STE que no eran equivalentes por trazas. La equivalencia por trazas-completas es mas fina que la de trazas, por lo tanto esos mismos STE no son equivalentes por trazas completas. Efectivamente $TC(\theta(\mathcal{G}_1)) = \{ab, ac\} \neq \{ab\} = TC(\theta(\mathcal{G}_2))$. Y esa misma prueba nos sirve para demostrar el teorema. ■

Teorema 3.3.3 *La semántica por failure no es una congruencia para los operadores de prioridad.*

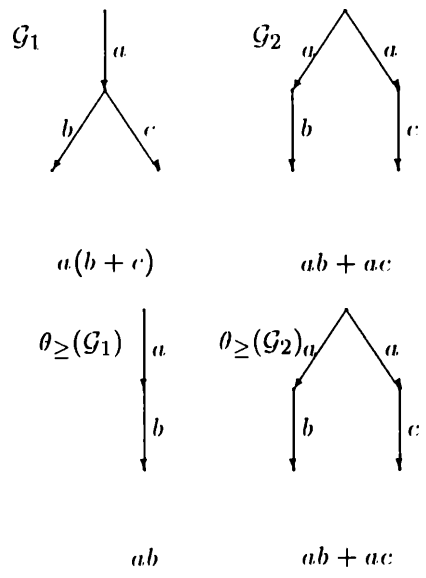


Figura 3.2: Trazas no es una congruencia para $\theta_{\geq}$.

Demostración: Sean $\mathcal{G}_1$ y $\mathcal{G}_2$ los STE de la figura 3.3. Sabemos por 2.7 que $\mathcal{G}_1 \equiv_{Re} \mathcal{G}_2$ entonces $\mathcal{G}_1 \equiv_F \mathcal{G}_2$. Definimos el orden parcial $\geq = \{f > c\}$ y obtenemos a partir de él los STE $\theta(\mathcal{G}_1)$ y $\theta(\mathcal{G}_2)$ de la misma figura.

Usamos el par failure $< ac, \{c\} >$ para demostrar que los STE resultantes no son failure equivalentes: se ve fácilmente que $< ac, \{c\} > \in F(\theta(\mathcal{G}_1)) \wedge < ac, \{c\} > \notin F(\theta(\mathcal{G}_2))$. Por lo tanto $\theta(\mathcal{G}_1) \not\equiv_F \theta(\mathcal{G}_2))$. ■

Usando los mismos STE de la demostración anterior obtenemos el resultado para la ready equivalencia.

Teorema 3.3.4 *La semántica por ready no es una congruencia para los operadores de prioridad.*

Demostración: Sabemos a partir de la definición de ready simulación que los STE $\mathcal{G}_1$ y $\mathcal{G}_2$ usados en el teorema 3.3.3 son ready equivalentes ($\mathcal{G}_1 \equiv_{Re} \mathcal{G}_2$). Al aplicar el θ del mismo teorema obtenemos los STE $\theta(\mathcal{G}_1)$ y $\theta(\mathcal{G}_2)$ de la figura 3.3). Vimos que esos STE no son failure equivalentes y por lo tanto tampoco son ready equivalentes ($\equiv_{Re} \subset \equiv_F$). Así se obtiene que $\theta(\mathcal{G}_1) \not\equiv_{Re} \theta(\mathcal{G}_2))$. ■

Obtuvimos un resultado similar para la failure trace equivalencia.

Teorema 3.3.5 *La semántica por failure trace no es una congruencia para los operadores de prioridad.*

Demostración: Tomemos los STE $\mathcal{G}_1$ y $\mathcal{G}_2$ de la figura 3.4. Se puede ver que $\mathcal{G}_1 \equiv_{FT} \mathcal{G}_2$. Definimos el orden parcial $\geq = \{f > b, d > c\}$ y obtenemos los $\theta(\mathcal{G}_1)$ y $\theta(\mathcal{G}_2)$ de la misma figura.

Finalmente usamos la failure trace $< a, \{c, b\} >< d, \emptyset >$ para demostrar que $\theta(\mathcal{G}_1) \not\equiv_{FT} \theta(\mathcal{G}_2)$, ya que $< a, \{c, b\} >< d, \emptyset > \notin FT(\theta(\mathcal{G}_1))$ y $< a, \{c, b\} >< d, \emptyset > \in FT(\theta(\mathcal{G}_2))$. ■

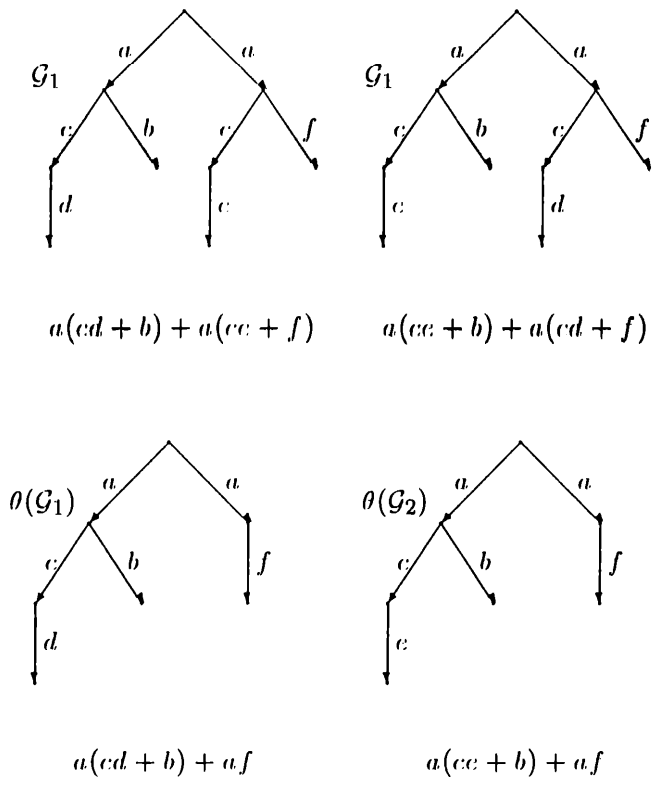


Figura 3.3: Failure no es una equivalencia para $\theta_{\geq}$.

Terminamos esta sección con el resultado de congruencia para la equivalencia por futuros posibles. Este también es un resultado negativo, similar a los anteriores.

Teorema 3.3.6 *La semántica por futuros posibles no es una congruencia para los operadores de prioridad.*

Demostración: Tomemos los STE $\mathcal{G}_1$ y $\mathcal{G}_2$ que se usaron en la figura 2.9. Sabemos que $\mathcal{G}_1 \equiv_{FP} \mathcal{G}_2$. Si consideramos el orden parcial $\geq = \{b > c\}$, a partir del cual obtenemos los $\theta(\mathcal{G}_1)$ y $\theta(\mathcal{G}_2)$ de la figura 3.5, encontramos que $< c, \{ab, ac\} > \notin FP(\theta(\mathcal{G}_1))$ y $< c, \{ab, ac\} > \in FP(\theta(\mathcal{G}_2))$. Por lo tanto $\theta(\mathcal{G}_1) \not\equiv_{FP} \theta(\mathcal{G}_2))$. ■

Para la simulación volvemos a tener un resultado negativo, esto es, los operadores de prioridad no son compatibles con la simulación.

Teorema 3.3.7 *La simulación no es una congruencia para los operadores de prioridad.*

Demostración: Consideremos los STE $\mathcal{G}_1$ y $\mathcal{G}_2$ de la figura 3.6. Podemos ver que $\mathcal{G}_1 \rightleftharpoons \mathcal{G}_2$.

Si tomamos el orden parcial $\geq = \{c > b\}$ sobre Act y aplicamos el operador de prioridad asociado a él obtenemos los $\theta(\mathcal{G}_1)$ y $\theta(\mathcal{G}_2)$ de la misma figura.

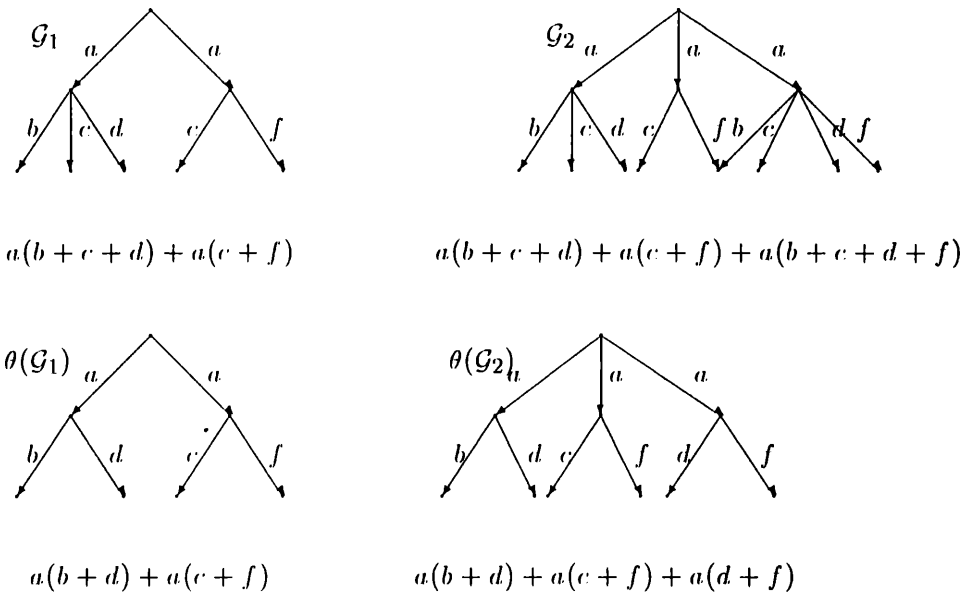


Figura 3.4: Failure trace no es una congruencia para $\theta_{\geq}$

Probamos que $\theta(\mathcal{G}_1) \not\rightleftharpoons \theta(\mathcal{G}_2)$ usando la fórmula HML $f = \langle a \rangle \langle b \rangle T \in \mathcal{L}_1$, ya que $\theta(\mathcal{G}_1) \not\models f$ y $\theta(\mathcal{G}_2) \models f$. ■

3.4 Equivalencias preservadas por el operador de prioridad

Nos dedicamos ahora a las equivalencias más finas del reticulado de semánticas. Ready-trace, ready simulación y bisimulación son congruencias para los operadores de prioridad. En la literatura se encuentra que la semántica por ready-trace es una congruencia para la clase de operadores de prioridad [BBK85], y que ninguna equivalencia más gruesa que esa lo es, tal como se vió en la sección anterior. En [BBK86] se introduce que los operadores de prioridad

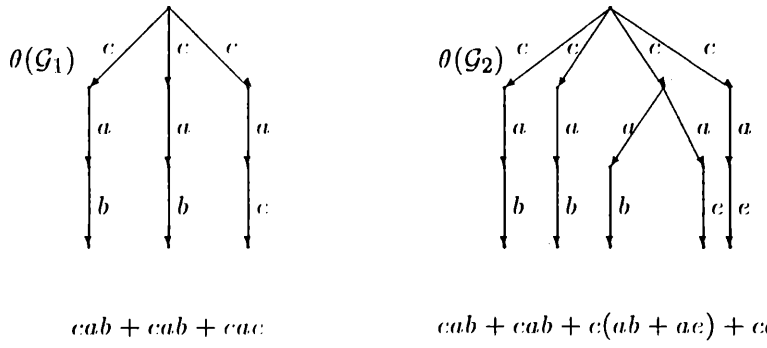


Figura 3.5: Futuros posibles no es una congruencia para los operadores de prioridad

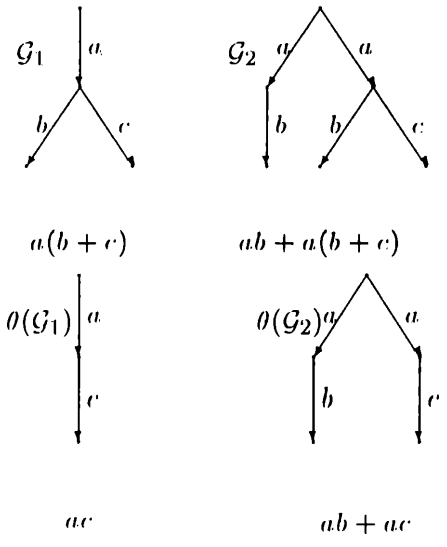


Figura 3.6: Simulación no es una congruencia para los operadores de prioridad

son compatibles con la bisimulación. En este trabajo se introduce el resultado con respecto a la ready-simulación: la ready-simulación es una congruencia con respecto a los operadores de prioridad.

En el siguiente lema damos una función f que a partir de un conjunto de ready traces y un orden parcial $\geq$ nos permite encontrar el conjunto de ready traces del STE al que se le aplicó el operador $\theta_{\geq}$. Esto nos va a ser escencial mas adelante para demostrar que la semántica por ready-trace es una congruencia para los operadores de prioridad.

Lema 3.4.1 *Dado* $\mathcal{G} = (S, i, Act, \longrightarrow) \in \mathcal{C}_{STE}$. *Existe una función* f *tal que* $f(RT(\mathcal{G})) = RT(\theta(\mathcal{G}))$.

Demostración: Sea $\mathcal{G} = (S, i, Act, \longrightarrow)$ un STE. Sin pérdida da generalidad usamos la forma normalizada para las secuencias del conjunto $RT(\mathcal{G})$ (ver [BBK85]), en donde $s =< \epsilon, X_0 > < a_1, X_1 > \ldots < a_n, X_n > \in RTN(\mathcal{G})$ sii:

- $X_0 = I(i)$;
- $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \ldots p_{n-1} \xrightarrow{a_n} p_n$, ($p_0\ a_1\ p_1\ a_2\ p_2 \ldots a_n\ p_n$ es un camino en $\mathcal{G}$);
- $p_0 = i$
- $X_j = I(p_j)$;
- $a_j \in X_{j-1}$;
- $X_n = \emptyset$.

A partir de ese conjunto normalizado de ready traces y un orden parcial $\geq$, definido sobre las acciones de Act, obtenemos $\theta(\mathcal{G}) = (S, i, Act, \longrightarrow^{\theta}) \in \mathcal{C}_{STE}$. Definimos la función parcial $fp : (Act \times P(Act))^* \longrightarrow (Act \times P(Act))^*$ de la siguiente manera:

$$fp(< \epsilon, X_0 >< a_1, X_1 > \ldots < a_n, X_n >) =< \epsilon, X_0' >< a_1, X_1' > \ldots < a_n, X_n' >$$

si existen X'_1, X'_n tales que: $X'_j = \{a : a \in X_j \wedge \forall b \in X_j : b \not> a\}$ y $a_j \in X'_{j-1}$. Si X es un conjunto de ready traces entonces definimos $f(X) = \{x' : x \in X \wedge x' = fp(x)\}$.

Tenemos que demostrar ahora que esa función calcula realmente el conjunto de ready traces de $\theta(\mathcal{G})$.

$(f(RT(\mathcal{G})) \subseteq RT(\theta(\mathcal{G})))$ Sea $s' =< \epsilon, X'_0 >< a_1, X'_1 > \ldots < a_n, X'_n > \in f(RTN(\mathcal{G}))$ entonces, por definición de f, $\exists s \in RT(\mathcal{G}) : s =< \epsilon, X_0 >< a_1, X_1 > \ldots < a_n, X_n >$, que forma un camino $c = p_0\ a_1\ p_1\ a_2\ p_2 \ldots a_n\ p_n$ en $\mathcal{G}$, con $X'_j \subseteq X_j$. Supongamos que ese camino c no es un camino de $\theta(\mathcal{G})$. Entonces $\exists j$ tal que $p_j \xrightarrow{b} q, b \in X_j$ y $b > a_{j+1}$, con lo cual $a_{j+1} \notin X'_j$, lo cual es absurdo porque en s' se cumple $a_{j+1} \in X_j$. Por lo tanto c es un camino de $\theta(\mathcal{G})$ y existe una ready-trace en $RT(\theta(\mathcal{G}))$ con esas acciones.

Veamos ahora que los conjuntos iniciales de esa ready-trace en $RT(\theta(\mathcal{G}))$ y en $f(RT(\mathcal{G}))$ son los mismos: Sea una acción $a \in X'_j \overset{\text{def}}{\Leftrightarrow} \forall b \in Act$ tal que $b \in X_j : b \not> a \Leftrightarrow a \in I_\theta(p_j)$.

Así obtenemos que $s' \in RT(\theta(\mathcal{G}))$.

$(RT(\theta(\mathcal{G})) \subseteq f(RT(\mathcal{G})))$ Sea $s' =< \epsilon, X'_0 >< a_1, X'_1 > \ldots < a_n, X'_n > \in RT(\theta(\mathcal{G}))$. Eso significa que $c = p_0\ a_1\ p_1\ a_2\ p_2 \ldots a_n\ p_n$ es un camino de $\theta(\mathcal{G})$. Por la definición 3.1.1 de θ, c es un camino en $\mathcal{G}$. Entonces podemos construir una ready-trace normalizada a partir de las acciones de ese camino: $s =< \epsilon, X_0 >< a_1, X_1 > \ldots < a_n, X_n >$, con $X_j = I_{\mathcal{G}}(p_j)$. Supongo que $s'' =< \epsilon, X''_0 >< a_1, X''_1 > \ldots < a_n, X''_n >= fp(s) \neq s'$. Como las ready-traces tienen las mismas acciones tienen que diferir en los conjuntos de inicios: $\exists j : X''_j \neq X'_j$. Sea $c \in X''_j \overset{\text{def}}{\Leftrightarrow} c \in X_j \wedge \forall b \in X_j : b \not> c \Leftrightarrow c \in I_\theta(p_j) \Leftrightarrow c \in X'_j$. Con lo cual $X''_j = X'_j$ y lo supuesto es absurdo: $s'' = f(s) = s'$.

Finalmente $f(RT(\mathcal{G})) = RT(\theta(\mathcal{G}))$. ■

El resultado que buscábamos con respecto a ready-trace:

Teorema 3.4.2 *La semántica por ready-trace es una congruencia para los operadores de prioridad.*

Demostración: Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$ tales que $\mathcal{G}_1 \equiv_{RT} \mathcal{G}_2$, entonces $RT(\mathcal{G}_1) = RT(\mathcal{G}_2) \Rightarrow f(RT(\mathcal{G}_1)) = f(RT(\mathcal{G}_2)) \Rightarrow$ (por el lema 3.4.1) $RT(\theta(\mathcal{G}_1)) = RT(\theta(\mathcal{G}_2))$. Por lo tanto $\theta(\mathcal{G}_1) \equiv_{RT} \theta(\mathcal{G}_2)$. ■

Vamos a demostrar ahora que la ready-simulación es una congruencia para la familia de operadores que estamos tratando. Para ello es necesario encontrar una ready-simulación que relacione a los STE obtenidos por la aplicación del operador de prioridad. En un primer paso vamos a demostrar que la misma R que relaciona a los STE originales es una ready-simulación para los nuevos procesos .

Teorema 3.4.3 *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$ y θ un operador de prioridad. Si $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$ entonces $R : \theta(\mathcal{G}_1) \subseteq_R \theta(\mathcal{G}_2)$.*

Demostración: Sean $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$ y pRq. De la definición 3.1.1 obtenemos que $I_{\theta(\mathcal{G}_1)}(p) = \{a : a \in I(p) \wedge \forall b \in I(p) : b \not> a\} =$ (por definición de ready-simulación) $\{a : a \in I(q) \wedge \forall b \in I(q) : b \not> a\} = I_{\theta(\mathcal{G}_2)}(q)$. Tomamos $p, p' \in S_1, a \in Act : p \xrightarrow{a}_{\theta(1)} p'$, entonces $p \xrightarrow{a}_1 p' \overset{\text{def}}{\Rightarrow} \exists q' \in S_2 : q \xrightarrow{a}_2 q' \wedge p'Rq'$. Si suponemos que $q \not\xrightarrow{a}_{\theta(2)} \Rightarrow \exists b \in I(q) : b > a$. Pero $b \in I(q) \Rightarrow b \in I(p)$. Por lo tanto $a \notin I_{\theta(\mathcal{G}_1)}(p)$, lo cual es absurdo. Así $q \xrightarrow{a}_{\theta(2)} q'$.

Finalmente obtenemos que $R : \theta(\mathcal{G}_1) \subseteq_R \theta(\mathcal{G}_2)$. ■

Corolario 3.4.4 *La ready-simulación es una congruencia para los operadores de prioridad.*

Finalmente vamos a demostrar que la bisimulación es una congruencia para nuestra familia de operadores. Nuevamente vamos a usar la misma relación R dada para mostrar que los STE resultantes de la aplicación del operador son bisimilares. Previo a la demostración vamos a mostrar que esa relación R preserva los conjuntos de inicios de los estados.

Lema 3.4.5 *Sean* $\mathcal{G}_1 = (S_1, i_1, Act, \longrightarrow_1), \mathcal{G}_2 = (S_2, i_2, Act, \longrightarrow_2) \in \mathcal{C}_{STE}$ *tales que* $R : \mathcal{G}_1 \leftrightarrows \mathcal{G}_2$, *y sean* $p \in S_1, q \in S_2$ *tales que* pRq. *Entonces* $I_{\mathcal{G}_1}(p) = I_{\mathcal{G}_2}(q)$ *y* $I_{\theta(\mathcal{G}_1)}(p) = I_{\theta(\mathcal{G}_2)}(q)$.

Demostración: Tenemos pRq. Tomamos $b \in I_{\mathcal{G}_1}(p) \Leftrightarrow \exists p' \in S_1 . p \xrightarrow{b}_1 p' \overset{\text{def}}{\Leftrightarrow} \exists q' : q \xrightarrow{b}_2 q' \Leftrightarrow b \in I_{\mathcal{G}_2}(q)$.

Además $I_{\theta(\mathcal{G}_1)}(p) = \{a : a \in I_{\mathcal{G}_1}(p) \wedge \forall b \in I_{\mathcal{G}_1}(p) : b \not> a\} = \{a : a \in I_{\mathcal{G}_2}(q) \wedge \forall b \in I_{\mathcal{G}_2}(q) : b \not> a\} = I_{\theta(\mathcal{G}_2)}(q)$. ■

Teorema 3.4.6 *La bisimulación es una congruencia para los operadores de prioridad.*

Demostración: Sea $R : \mathcal{G}_1 \leftrightarrows \mathcal{G}_2$ tal que pRq y sean $p' \in S_1, a \in Act : p \xrightarrow{a}{}^{\theta}_1 p'$. Entonces, por la definición 3.1.1 de los operadores de prioridad, $p \xrightarrow{a}_1 p'$. Como R es una bisimulación: $\exists q' \in S_2 : q \xrightarrow{a}_2 q'$, con $p'Rq'$.

Sabemos que $a \in I(q)$ porque $a \in I(p)$. Supongamos que $q \not\xrightarrow{a}{}^{\theta}_2 \Rightarrow \exists b \in I(q) : b > a$. Pero $b \in I(q) \Rightarrow$ (lema 3.4.5) $b \in I(p)$. Por lo tanto $a \notin I_\theta(p)$. Absurdo. Así $q \xrightarrow{a}{}^{\theta}_2 q'$.

De la misma manera se demuestra que $pRq \wedge q \xrightarrow{a}{}^{\theta}_2 q' \Rightarrow p \xrightarrow{a}{}^{\theta}_1 p' \wedge p'Rq'$. Finalmente $\theta(\mathcal{G}_1) \leftrightarrows \theta(\mathcal{G}_2)$. ■

Capítulo 4

El caso de la n-nested simulación

4.1 La n-nested simulación y los operadores de prioridad

El caso de la n-nested simulación presentó resultados inesperados y fué objeto de un estudio más profundo. La n-nested simulación se encuentra entre la ready-simulación y la bisimulación en el reticulado de equivalencias semánticas. Estas dos equivalencias son congruencias para los operadores de prioridad. Por eso nuestra intuición nos decía que la n-nested simulación también debía ser una congruencia para esos operadores. La n-nested simulación tiene, al menos, tanta información sobre los procesos como la ready-simulación, y nos parecía que eso alcanzaba para que se comportara "correctamente" con respecto a los operadores de prioridad. Veremos, sin embargo, que la n-nested simulación no es una congruencia para esa clase de operadores. A partir de este resultado sabemos que las n-nested simulaciones no pueden ser usadas en modelos en los cuales la prioridad tenga sentido.

Para demostrar que las n-nested simulaciones no son congruencias para los operadores estudiados damos un par de sistemas n-nested similares y un operador de prioridad, tal que al aplicarlo a los sistemas obtenemos nuevos sistemas que no son n-nested similares. Hacemos esto para todo natural n.

En nuestro teorema vamos a necesitar el siguiente lema: Damos una familia de pares de sistemas a partir de los términos s'_n y t'_n y una familia de fórmulas HML, de a lo sumo $n-1$ negaciones anidadas, que demuestran que esos STE no son n-nested similares.

Lema 4.1.1 *Sea $f_n \in \mathcal{L}_n$ la familia de fórmulas definidas por: $f_1 = \langle a\rangle\langle b\rangle T$ y $f_{n+1} = \langle a\rangle\neg f_n$, y sean los términos s'_n y t'_n dados por:*

$$\begin{array}{ll} s'_0 = b & t'_0 = c \\ s'_{n+1} = a.t'_n & t'_{n+1} = a.s'_n + a.t'_n \end{array}$$

Entonces $t'_n \models f_n$ y $s'_n \not\models f_n$.

Demostración: Usamos inducción sobre n.

(**caso** $n = 2$) Los valores de s'_2, t'_2 y f_2 respectivamente son los siguientes:
$s'_2 = a.t'_1 = a.(a.s'_0 + a.t'_0) = a.(a.b + a.c)$
$t'_2 = a.s'_1 + a.t'_1 = a.a.t'_0 + a.(a.s'_0 + a.t'_0) = a.a.c + a.(a.b + a.c)$

$f_2 = \langle a\rangle\neg\langle a\rangle\langle b\rangle T$,

que corresponden con los STE de la figura 4.1.

Tenemos que probar que $a.(a.b + a.c) \not\models \langle a\rangle\neg\langle a\rangle\langle b\rangle T$ y $a.a.c + a.(a.b + a.c) \models \langle a\rangle\neg\langle a\rangle\langle b\rangle T$:

$a.(a.b + a.c) \not\models \langle a\rangle\neg\langle a\rangle\langle b\rangle T \Leftrightarrow a.b + a.c \not\models \neg\langle a\rangle\langle b\rangle T \Leftrightarrow a.b \not\models \neg\langle a\rangle\langle b\rangle T \vee a.c \not\models \neg\langle a\rangle\langle b\rangle T \Leftarrow a.b \models \langle a\rangle\langle b\rangle T \Leftrightarrow b \models \langle b\rangle T$

$a.a.c + a.(a.b + a.c) \models \langle a\rangle\neg\langle a\rangle\langle b\rangle T \Leftrightarrow a.a.c \models \langle a\rangle\neg\langle a\rangle\langle b\rangle T \vee a.(a.b + a.c) \models \langle a\rangle\neg\langle a\rangle\langle b\rangle T \Leftarrow a.a.c \models \langle a\rangle\neg\langle a\rangle\langle b\rangle T \Leftrightarrow a.c \models \neg\langle a\rangle\langle b\rangle T \Leftrightarrow a.c \not\models \langle a\rangle\langle b\rangle T \Leftrightarrow c \not\models \langle b\rangle T$

(caso $n > 2$) Tomamos como hipótesis inductiva $s'_{n-1} \not\models f_{n-1}$ y $t'_{n-1} \models f_{n-1}$ y tenemos que probar que $s'_n \not\models f_n$ y $t'_n \models f_n$.

$s'_n \not\models f_n \Leftrightarrow a.t'_{n-1} \not\models \langle a\rangle\neg f_{n-1} \Leftrightarrow t'_{n-1} \not\models \neg f_{n-1} \Leftrightarrow t'_{n-1} \models f_{n-1}$

$t'_n \models f_n \Leftrightarrow a.s'_{n-1} + a.t'_{n-1} \models \langle a\rangle\neg f_{n-1} \Leftrightarrow a.s'_n \models \langle a\rangle\neg f_n \vee a.t'_n \models \langle a\rangle\neg f_n \Leftarrow a.s'_n \models \langle a\rangle\neg f_n \Leftrightarrow s'_{n-1} \models \neg f_{n-1} \Leftrightarrow s'_{n-1} \not\models f_{n-1}$ ■

Ahora damos nuevos pares de sistemas que sabemos son n-nested similares y construimos un operador de prioridad tal que al aplicarlo a los sistemas obtenemos los procesos del lema 4.1.1. Así obtenemos que las n-nested simulaciones no tienen un comportamiento adecuado con respecto a esta clase de operadores.

Los pares de sistemas a los que nos referimos son los $\mathcal{G}_1^n$ y $\mathcal{G}_2^n$ construidos a partir de los términos s_n y t_n del ejemplo 2.3.13. Sabemos que son n-nested similares a partir de [GV89], pero igualmente lo demostramos dando relaciones R y Q tales que $R, Q : \mathcal{G}_1^n \rightleftharpoons_n \mathcal{G}_2^n$, esto es, $R : \mathcal{G}_1^n \subseteq_{\rightarrow n} \mathcal{G}_2^n$ y $Q : \mathcal{G}_2^n \subseteq_{\rightarrow n} \mathcal{G}_1^n$.

Vamos a construir esas relaciones a partir de las simulaciones obtenidas en el siguiente lema. Nos abusamos de la notación, usando los términos s_i y t_i como estados pertenecientes a S. El término indica exactamente las acciones que puede realizar el estado.

Lema 4.1.2 *Sean $\mathcal{G}_1, \mathcal{G}_2$ los STE correspondientes a los términos s_n y t_n respectivamente y sean*

$$R_n = Id \cup \{(s_n, t_n)\}$$

$$R_i = (R_{i+1})^{-1} \cup \{(s_i, t_i)\}$$

con $0 \leq i \leq n-1$. Entonces $R_{n-2j} : \mathcal{G}_1 \subseteq_{\rightarrow} \mathcal{G}_2$ y $R_{n-2j-1} : \mathcal{G}_2 \subseteq_{\rightarrow} \mathcal{G}_1$, con $0 \leq j \leq [n/2]$.

Demostración: (caso $j = 0$)

Queremos ver que $R_{n-0} : \mathcal{G}_1 \subseteq_{\rightarrow} \mathcal{G}_2$ y que $R_{n-0-1} : \mathcal{G}_2 \subseteq_{\rightarrow} \mathcal{G}_1$. $R_n = Id \cup \{(s_n, t_n)\}$.

Imaginemos una relación que permita a $\mathcal{G}_2$ simular a $\mathcal{G}_1$. Necesitamos incluir el par (s_n, t_n). Efectivamente podemos incluirlo ya que a partir de la definición de los términos vemos que $t_n = a.s_{n-1} + a.t_{n-1}$, con lo cual puede simular cualquier movimiento de $s_n = a.t_{n-1}$ (si $n = 1$ también lo simula: $t_n = b + c$ y $s_n = b$). Sólo es necesario incluir en la realción los pares $(a.t_{n-1}, a.t_{n-1}), (t_{n-1}, t_{n-1})$, etc, todos los cuales están incluidos en la realción Id (identidad). Esa relación que construímos es exactamente $R_n = Id \cup \{(s_n, t_n)\}$. Así $R_n : \mathcal{G}_1 \subseteq_{\rightarrow} \mathcal{G}_2$.

Si consideramos ahora $R_{n-1} = Id \cup \{(t_n, s_n), (s_{n-1}, t_{n-1})\}$. Vemos fácilmente, después del razonamiento anterior, que t_{n-1} simula a s_{n-1}. No queda tan claro que s_n simule a t_n. Imaginemos que t_n decide realizar las acciones $a.t_{n-1}$, no se presenta ningún problema para que s_n las imite ya que $s_n = a.t_{n-1}$. Si, en cambio, t_n elige ejecutar $a.s_{n-1}$ entonces s_n

también puede imitarlas ya que $s_n = a.t_{n-1}$, s_n imita la acción a ejecutando una acción a y vimos en el par anterior que t_{n-1} imita a s_{n-1}. Así obtenemos $R_{n-1} : \mathcal{G}_2 \subseteq_{\rightarrow} \mathcal{G}_1$.

(caso $j > 0$)

Como hipótesis inductiva tenemos que $R_{n-2k} : \mathcal{G}_1 \subseteq_{\rightarrow} \mathcal{G}_2$ y $R_{n-2k-1} : \mathcal{G}_2 \subseteq_{\rightarrow} \mathcal{G}_1$, con $k < j$. Queremos saber que sucede con R_{n-2j} y R_{n-2j-1}.

Por definición de R_n tenemos $R_{n-2j} = (R_{n-2j+1})^{-1} \cup \{(s_{n-2j}, t_{n-2j})\}$ y $R_{n-2j+1} = (R_{n-2j+1+1})^{-1} \cup \{(s_{n-2j+1}, t_{n-2j+1})\}$. Entonces $R_{n-2j} = R_{n-2j+1+1} \cup \{(t_{n-2j+1}, s_{n-2j+1}), (s_{n-2j}, t_{n-2j})\}$.

Pero $n-2j+1+1 = n-2(j-1) = n-2k, k < j$. Entonces, por hipótesis inductiva, sabemos que $R_{n-2j+1+1} : \mathcal{G}_1 \subseteq_{\rightarrow} \mathcal{G}_2$. A esa relación, que contiene el par (s_n, t_n) (porque $\mathcal{G}_2$ simula a $\mathcal{G}_1$), le podemos agregar nuevos pares y seguir teniendo una simulación. Es posible agregar el par (s_{n-2j}, t_{n-2j}), ya que t_{n-2j} simula a s_{n-2j}. Lo mismo sucede con el par (t_{n-2j+1}, s_{n-2j+1}), aunque vimos que no es tan inmediato ver que s_{n-2j+1} simula a t_{n-2j+1}. La nueva relación es R_{n-2j} y entonces $R_{n-2j} : \mathcal{G}_1 \subseteq_{\rightarrow} \mathcal{G}_2$.

De la misma forma procedemos con R_{n-2j-1}. Tenemos que $R_{n-2j-1} = R_{n-2j-1+1+1} \cup \{(t_{n-2j-1+1}, s_{n-2j-1+1}), (s_{n-2j-1}, t_{n-2j-1})\}$. Sabemos que $R_{n-2j-1+1+1} = R_{n-2k-1}$, con $k = j-1 < j$. Por hipótesis inductiva sabemos que $R_{n-2j-1+1+1} : \mathcal{G}_2 \subseteq_{\rightarrow} \mathcal{G}_1$ y al agregarle los dos pares redundantes obtenemos que $R_{n-2j-1} : \mathcal{G}_2 \subseteq_{\rightarrow} \mathcal{G}_1$. ■

A partir de esas simulaciones vamos a construir las n-nested simulaciones que relacionan a $\mathcal{G}_1^n$ y $\mathcal{G}_2^n$. Obtuvimos exactamente $n+1$ simulaciones, de R_0 a R_n, con el tipo de inclusión que necesitamos: $R_i \subseteq (R_{i-1})^{-1}$.

Lema 4.1.3 ($R, Q : \mathcal{G}_1^n \rightleftharpoons_n \mathcal{G}_2^n$) *Sean $\mathcal{G}_1$ y $\mathcal{G}_2$ los sistemas correspondientes a los términos s_n y t_n respectivamente y sean $R = R_n$ y $Q = R_{n-1}$. Entonces $R, Q : \mathcal{G}_1 \rightleftharpoons_n \mathcal{G}_2$.*

Demostración: Vimos en el lema anterior que $R_{n-2j} : \mathcal{G}_1 \subseteq_{\rightarrow} \mathcal{G}_2, 0 \leq j < [n/2]$. Además encontramos $R_{n-2j-1} : \mathcal{G}_2 \subseteq_{\rightarrow} \mathcal{G}_1$ tal que $R_{n-2j} \subseteq (R_{n-2j-1})^{-1}$. Desde R_{n-2j} hasta R_1 tenemos exactamente $n-2j$ relaciones anidadas, por lo tanto $R_{n-2j} : \mathcal{G}_1 \subseteq_{\rightarrow n-2j} \mathcal{G}_2$.

En particular, cuando $j = 0$ tenemos $R = R_n : \mathcal{G}_1 \subseteq_{\rightarrow n} \mathcal{G}_2$.

De la misma forma, si consideramos $R_{n-2j-1} : \mathcal{G}_2 \subseteq_{\rightarrow} \mathcal{G}_1, 0 \leq j < [n/2]$, existe $R_{n-2j-1-1} = R_{n-2k}, 1 \leq k < [n/2]+1$, con $R_{n-2j-1} \subseteq (R_{n-2j-1-1})^{-1}$. Vemos que desde R_{n-2j-1} hasta R_0 hay exactamente $n-2j-1+1 = n-2j$ simulaciones anidadas. Así obtenemos que $R_{n-2j-1} : \mathcal{G}_2 \subseteq_{\rightarrow n-2j} \mathcal{G}_1$.

En particular, cuando $j = 0$ tenemos $Q = R_{n-1} : \mathcal{G}_2 \subseteq_{\rightarrow n} \mathcal{G}_1$.

Finalmente $R, Q : \mathcal{G}_1 \rightleftharpoons_n \mathcal{G}_2$. ■

Volvamos entonces al tema que nos interesaba.

Teorema 4.1.4 (n-nested no es una congruencia para los operadores de prioridad) *Existen $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}, \theta : \mathcal{C}_{STE} \longrightarrow \mathcal{C}_{STE}$ tales que $\mathcal{G}_1 \rightleftharpoons_n \mathcal{G}_2 \wedge \theta(\mathcal{G}_1) \not\rightleftharpoons_n \theta(\mathcal{G}_2)$.*

Demostración: Usamos los STE $\mathcal{G}_1^n$ y $\mathcal{G}_2^n$ del ejemplo 2.3.13, que son n-nested similares. Elegimos θ con el orden parcial $\geq = \{c > b\}$. Vemos que $\theta(\mathcal{G}_1^n)$ y $\theta(\mathcal{G}_2^n)$ (ver figura 4.2) corresponden a los términos s'_n y t'_n del lema 4.1.1.

Consideramos también la familia de fórmulas $f_n \in \mathcal{L}_n$ del lema 4.1.1 y obtenemos que $\theta(\mathcal{G}_2) \models f_n \in \mathcal{L}_n$ y $\theta(\mathcal{G}_1) \not\models f_n \in \mathcal{L}_n$. ■

De éste teorema concluimos que la n-nested simulación no puede ser usada como equivalencia semántica en modelos en los que se usen operadores de prioridad. A partir de este resultado inesperado nuestra investigación puede tomar distintos rumbos. En un primer momento nos interesa estudiar un poco mas el comportamiento de las n-nested simulaciones en cuanto a estos operadores. Encontramos que se puede acotar su comportamiento, asegurando que al aplicar un operador de prioridad a STE n-nested similares obtenemos STE $n-1$-nested similares. Para ello necesitamos una serie de resultados:

Dos estados n-nested similares pueden realizar exactamente las mismas acciones. Es decir, sus conjuntos de acciones iniciales son iguales.

Lema 4.1.5 *Sean* $\mathcal{G}_1 = (S_1, i_1, Act, \longrightarrow_1), \mathcal{G}_2 = (S_2, i_2, Act, \longrightarrow_2) \in \mathcal{C}_{STE}$ *tales que* $R : \mathcal{G}_1 \subseteq_n \mathcal{G}_2, n \geq 2$, *y sean* $p \in S_1, q \in S_2$ *tales que* pRq. *Entonces* $I(p) = I(q)$.

Demostración: Veamos que $I(p) \subseteq I(q)$ y $I(q) \subseteq I(p)$.

(caso $\subseteq$) Tenemos $R : \mathcal{G}_1 \subseteq_n \mathcal{G}_2, pRq$ y tomamos $a \in I(p)$. Que a sea una acción inicial de p significa que $\exists p' \in S_1 : p \xrightarrow{a}_1 p'$. Por la definición de n-nested simulación sabemos que R es una simulación. Entonces $\exists q' \in S_2 : q \xrightarrow{a}_2 q'$. De allí obtenemos que $a \in I(q)$. Entonces $I(p) \subseteq I(q)$.

(caso $\supseteq$) Sabemos que $R : \mathcal{G}_1 \subseteq_n \mathcal{G}_2$ entonces, por la definición de $\rightleftharpoons_n$, $\exists Q : \mathcal{G}_2 \subseteq_{n-1} \mathcal{G}_1$, con $R^{-1} \subseteq Q$. Como $n \geq 2$ estamos seguros de que Q es al menos una simulación: $Q : \mathcal{G}_2 \subseteq \mathcal{G}_1$. A partir de pRq, obtenemos que qQp, y si tomamos $a \in I(q)$, que implica $\exists q' : q \xrightarrow{a}_2 q'$, obtenemos a partir de la definición de simulación que $\exists p' \in S_1 : p \xrightarrow{a}_1 p'$. Así $a \in I(p)$, y por lo tanto $I(q) \subseteq I(p)$.

Finalmente $I(p) = I(q)$. ■

A partir del lema anterior y la definición de n-nested simulación, que da como condición que la relación sea una simulación, se obtiene directamente:

Lema 4.1.6 (Una n-nested simulación R es una ready-simulación) *Dados* $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$. *Si* $R : \mathcal{G}_1 \subseteq_n \mathcal{G}_2$, $n \geq 2$, *entonces* $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$.

Es importante tener en cuenta que aplicando cualquier operador de prioridad a pares de STE n-nested similares se obtienen STE similares y ready-similares.

Lema 4.1.7 *Sean* $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$. $R : \mathcal{G}_1 \subseteq_n \mathcal{G}_2, n \geq 2 \Rightarrow R : \theta(\mathcal{G}_1) \subseteq_R \theta(\mathcal{G}_2)$ *y además* $R : \theta(\mathcal{G}_1) \subseteq \theta(\mathcal{G}_2)$.

Demostración: Si $R : \mathcal{G}_1 \subseteq_n \mathcal{G}_2$ entonces, por el lema 4.1.6, tenemos $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$. Ademas, por el teorema 3.4.3, $R : \theta(\mathcal{G}_1) \subseteq_R \theta(\mathcal{G}_2)$. $R : \theta(\mathcal{G}_1) \subseteq \theta(\mathcal{G}_2)$ sale inmediatamente de la definición de ready simulación, ya que toda ready-simulación es una simulación. ■

El siguiente teorema es el que restringe el comportamiento de θ.

Teorema 4.1.8 *Sean* $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$, *tales que* $R : \mathcal{G}_1 \subseteq_{n+1} \mathcal{G}_2, n \geq 1$. ***Entonces, para todo operador de prioridad*** θ, *se cumple que* $R : \theta(\mathcal{G}_1) \subseteq_n \theta(\mathcal{G}_2)$.

Demostración: Vamos a probar que la misma relación R hace n-nested similares a $\theta(\mathcal{G}_1)$ y $\theta(\mathcal{G}_2)$.

(caso $n = 1$) Directamente del lema 4.1.7 sabemos que $R : \mathcal{G}_1 \subseteq_2 \mathcal{G}_2$ implica $R : \theta(\mathcal{G}_1) \subseteq \theta(\mathcal{G}_2)$, que es lo mismo que $R : \theta(\mathcal{G}_1) \rightleftharpoons_1 \theta(\mathcal{G}_2)$.

(caso $n > 1$) Sea $R : \mathcal{G}_1 \subseteq_{n+1} \mathcal{G}_2$. Queremos ver que $R : \theta(\mathcal{G}_1) \subseteq_n \theta(\mathcal{G}_2)$. Para ello usamos la definición de n-nested simulación y probamos que $R : \theta(\mathcal{G}_1) \subseteq \theta(\mathcal{G}_2)$ y que $\exists Q : \mathcal{G}_2 \subseteq_{n-1} \mathcal{G}_1$, con $R^{-1} \subseteq Q$.

A partir del lema 4.1.7 sabemos que $R : \theta(\mathcal{G}_1) \subseteq \theta(\mathcal{G}_2)$. Además, por definición de n-nested simulación tenemos que $\exists Q : \mathcal{G}_2 \subseteq_n \mathcal{G}_1 \wedge R^{-1} \subseteq Q$, y que por hipótesis inductiva $Q : \theta(\mathcal{G}_2) \subseteq_{n-1} \theta(\mathcal{G}_1)$. Por lo tanto $R : \theta(\mathcal{G}_1) \subseteq_n \theta(\mathcal{G}_2)$. ■

El teorema nos da una idea mas precisa del comportamiento de la clase de operadores de prioridad en el marco de las n-nested simulaciones. Este comportamiento está limitado por las $n-1$-nested simulaciones, esto es, si tenemos dos STE que son n-nested equivalentes podemos asegurar que al aplicarles un operador de prioridad obtenemos STE $n-1$-nested equivalentes.

Corolario 4.1.9 (Comportamiento de θ con respecto a la n-nested simulación) *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$. Para $n \geq 1 : \mathcal{G}_1 \rightleftharpoons_{n+1} \mathcal{G}_2$ implica $\theta(\mathcal{G}_1) \rightleftharpoons_n \theta(\mathcal{G}_2)$.*

Inmediatamente nos preguntamos que sucede con la inversa de este corolario. Esto es, podemos decir que $\mathcal{G}_1$ y $\mathcal{G}_2$ son $n+1$-nested similares si sabemos que $\mathcal{G}_1 \rightleftharpoons_n \mathcal{G}_2$ y que para cualquier operador de prioridad θ: $\theta(\mathcal{G}_1) \rightleftharpoons_n \theta(\mathcal{G}_2)$?

La respuesta es que no y lo vamos a demostrar dando un par de STE que no son $n+1$-nested similares y sin embargo al aplicarles cualquier operador de prioridad se obtienen STE n-nested similares. En este primer lema tenemos pares de STE n-nested similares y demostramos que no son $n+1$-nested similares.

Lema 4.1.10 *Sean $\mathcal{G}_1^n, \mathcal{G}_2^n \in \mathcal{C}_{STE}$ (ver figura 4.3) los STE correspondientes a los términos u_n y v_n.*

$$\begin{array}{lll} u_0 = a & & v_0 = a + aa \\ u_{m+1} = a.v_m & & v_{m+1} = a.u_m + a.v_m \end{array}$$

Entonces $\mathcal{G}_1^n \not\rightleftharpoons_{n+1} \mathcal{G}_2^n$.

Demostración: Consideramos los STE dados y la familia $f_n \in \mathcal{L}_n$ de fórmulas HML definida de la siguiente manera: $f_1 = \langle a \rangle \langle a \rangle T$ y $f_{n+1} = \langle a \rangle \neg f_n$. Vemos que $\mathcal{G}_2^n \models f_{n+1} \in \mathcal{L}_{n+1}$ y $\mathcal{G}_1^n \not\models f_{n+1} \in \mathcal{L}_{n+1}$, con lo cual $\mathcal{G}_1^n \not\rightleftharpoons_{n+1} \mathcal{G}_2^n$.

(caso $n = 1$) $\mathcal{G}_1^1 \not\models f_2 \Leftrightarrow a(a+aa) \not\models \langle a \rangle \neg \langle a \rangle \langle a \rangle T \Leftrightarrow a + aa \not\models \neg \langle a \rangle \langle a \rangle T \Leftrightarrow a + aa \models \langle a \rangle \langle a \rangle T \Leftrightarrow a \models \langle a \rangle \langle a \rangle T \vee aa \models \langle a \rangle \langle a \rangle T \Leftarrow aa \models \langle a \rangle \langle a \rangle T \Leftrightarrow a \models \langle a \rangle T$.

$\mathcal{G}_2^1 \models f_2 \Leftrightarrow a(a+aa)+aa \models \langle a \rangle \neg \langle a \rangle \langle a \rangle T \Leftrightarrow a(a+aa) \models \langle a \rangle \neg \langle a \rangle \langle a \rangle T \vee aa \models \langle a \rangle \neg \langle a \rangle \langle a \rangle T \Leftarrow aa \models \langle a \rangle \neg \langle a \rangle \langle a \rangle T \Leftrightarrow a \models \neg \langle a \rangle \langle a \rangle T \Leftrightarrow a \not\models \langle a \rangle \langle a \rangle T \Leftrightarrow 0 \not\models \langle a \rangle T$.

(caso $n > 1$) Supongo que $\mathcal{G}_2^{n-1} \models f_n$ y $\mathcal{G}_1^{n-1} \not\models f_n$ y quiero ver que $\mathcal{G}_2^n \models f_{n+1}$ y $\mathcal{G}_1^n \not\models f_{n+1}$.

$\mathcal{G}_1^n \not\models f_{n+1} \Leftrightarrow a\mathcal{G}_2^{n-1} \not\models \langle a \rangle \neg f_n \Leftrightarrow \mathcal{G}_2^{n-1} \not\models \neg f_n \Leftrightarrow \mathcal{G}_2^{n-1} \models f_n$, que vale por hipótesis.

$\mathcal{G}_2^n \models f_{n-1} \Leftrightarrow a\mathcal{G}_1^{n-1} + a\mathcal{G}_2^{n-1} \models \langle a \rangle \neg f_n \Leftrightarrow a\mathcal{G}_1^{n-1} \models \langle a \rangle \neg f_n \vee a\mathcal{G}_2^{n-1} \models \langle a \rangle \neg f_n \Leftarrow a\mathcal{G}_1^{n-1} \models \langle a \rangle \neg f_n \Leftrightarrow \mathcal{G}_1^{n-1} \models \neg f_n \mathcal{G}_1^{n-1} \not\models f_n$, cierto por hipótesis. ■

Lo interesante de los pares de STE definidos en el lema anterior es que esos sistemas permanecen intactos ante la aplicación de cualquier operador de prioridad ya que en ningún momento hay que elegir entre acciones diferentes.

Nota 4.1.11 (La inversa del corolario 4.1.9 no es cierta) *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$ tales que para cualquier operador θ : $\theta(\mathcal{G}_1) \rightleftharpoons_n \theta(\mathcal{G}_2)$. Eso no necesariamente implica que $\mathcal{G}_1 \rightleftharpoons_{n+1} \mathcal{G}_2$.*

Demostración: Para demostrar ésta nota vamos a dar pares de STE tales que son n-nested similares para cualquier operador $\theta_{\geq}$ definido. Luego vamos a demostrar que esos STE no son $n+1$-nested equivalentes.

Consideramos los STE $\mathcal{G}_1^n$ y $\mathcal{G}_2^n$ obtenidos a partir de los términos s_n y t_n del ejemplo 2.3.13, en dónde sólo cambiamos $s_0 = b$ y $t_0 = c$ por $s_0 = a$ y $t_0 = aa$ respectivamente. Los cambios realizados en s_0 y t_0 se pueden trasladar a las relaciones R y Q con las que demostramos que esos STE eran n-nested similares, con lo cual sabemos que los nuevos STE también lo son. En la figura 4.4 podemos ver un ejemplo para $n = 2$.

Sabemos que estos nuevos STE son n-nested similares (podemos escribir relaciones del tipo de las usadas en el ejemplo 2.3.12). Además coinciden con los STE del lema 4.1.10, con lo cual no son $n+1$-nested similares. Se ve fácilmente que para todo θ, definido a partir de cualquier orden parcial en las acciones de Act se cumple que $\theta(\mathcal{G}_1^n) = \mathcal{G}_1^n \rightleftharpoons_n \mathcal{G}_2^n = \theta(\mathcal{G}_2^n)$ y sabemos que $\theta(\mathcal{G}_1^n) = \mathcal{G}_1^n \not\rightleftharpoons_{n+1} \mathcal{G}_2^n = \theta(\mathcal{G}_2^n)$. ■

4.2 n-nested congruencia con respecto a θ

Vimos que la n-nested simulación no es una congruencia para la clase de los operadores de prioridad, entonces es natural que nos preguntemos por la mayor congruencia para estos operadores incluida en la n-nested simulación. Esto es, si nos ubicamos en el reticulado de las equivalencia semánticas a la altura de la n-nested simulación y nos dirigimos hacia las equivalencias mas finas: cuál es la primer equivalencia que es una congruencia para estos operadores? En esta sección la vamos a definir y le vamos a dar el nombre de n-nested congruencia. Sabemos que la bisimulación es mas fina que la n-nested simulación y además es una congruencia para los operadores de prioridad. Es posible que la n-nested congruencia coincida con la bisimulación. En caso de que esa equivalencia no coincida con alguna equivalencia conocida nos interesa estudiarla más profundamente y obtener su caracterización relacional o lógica.

Definición 4.2.1 (n-nested congruencia) *Definimos la* n-nested congruencia, *notación $\sim_n^\theta$, como la mayor congruencia para los operadores de prioridad incluida en la n-nested simulación. Esto es, para todo θ: $\mathcal{G}_1 \sim_n^\theta \mathcal{G}_2 \Rightarrow \theta(\mathcal{G}_1) \sim_n^\theta \theta(\mathcal{G}_2)$.*

De la definición anterior obtenemos $\sim_n^\theta \subseteq \rightleftharpoons_n$, y del teorema 4.1.4 vemos que $\sim_n^\theta \neq \rightleftharpoons_n$, con lo cual $\sim_n^\theta \subset \rightleftharpoons_n$.

A continuación vamos a descartar la posibilidad de que la n-nested congruencia coincida con la bisimulación: Consideremos los STE de la figura 4.4. Sabemos que son n-nested similares y además para todo θ: $\theta(\mathcal{G}_1) \rightleftharpoons_2 \theta(\mathcal{G}_2)$. Entonces $\mathcal{G}_1 \sim_2^\theta \mathcal{G}_2$. Definitivamente esos STE no son bisimilares porque en el lema 4.1.10 probamos que ni siquiera son 3-nested similares. A continuación terminamos de ubicar a la n-nested congruencia en el reticulado de las equivalencias semánticas (ver figura 6.1). Al tema de la caracterización de la equivalencia se va a volver más adelante.

Teorema 4.2.2 (n-nested congruencia en el reticulado semántico) $\rightleftharpoons_{n+1} \subset \sim_n^{\emptyset} \subset \rightleftharpoons_n$.

Demostramos este teorema para un conjunto bastante más general de operadores en el lema 5.3.14.

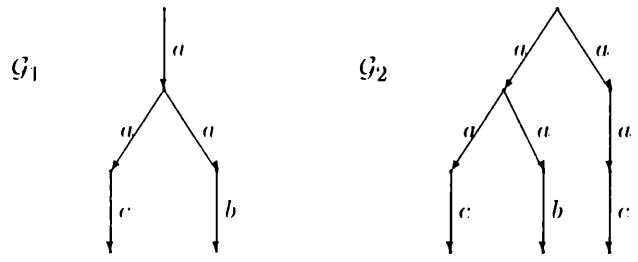


Figura 4.1: STE 2-nested similares

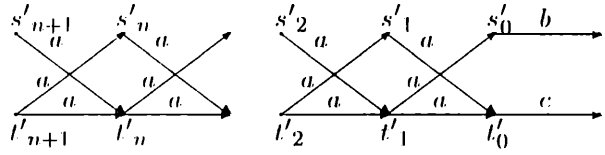


Figura 4.2: STE resultado de aplicar ∂ a $\mathcal{G}_1$ y a $\mathcal{G}_2$

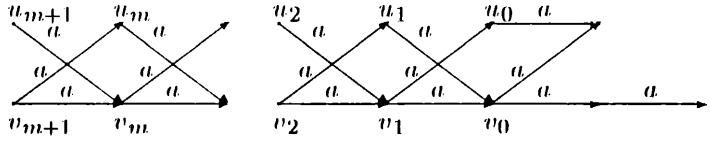


Figura 4.3: STE correspondientes a los términos u_n y v_n

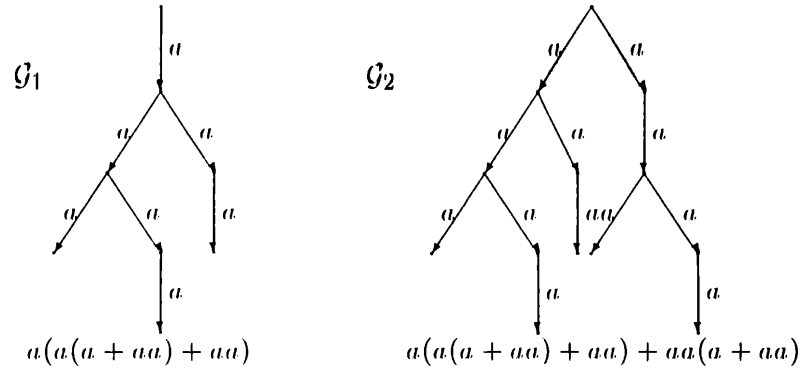


Figura 4.4: STE correspondientes a u_2 y v_2

Capítulo 5

U: Una clase de operadores estáticos unarios

En el capítulo anterior nos encontramos con un comportamiento inesperado de las n-nested simulaciones con respecto a los operadores de prioridad. Nos concentramos en limitar el comportamiento de estos operadores en el marco de esta familia de equivalencias pero no tuvimos en cuenta que pasa con otros operadores. Es posible que el caso de los operadores de prioridad sea un caso aislado o que existan más operadores en las mismas condiciones. Es por eso que decidimos ampliar la clase de los operadores a estudiar y consideramos un conjunto mucho más amplio. Entre ellos se encuentran muchos de los operadores que más aparecen en la literatura de sistemas reactivos. Nos gustaría poder exactamente cuales de esos operadores son compatibles con las n-nested simulaciones.

5.1 Definición de U

Definimos entonces una clase bastante general de operadores: Tomamos a los operadores unarios estáticos, U, sobre los STE. Con operadores estáticos nos referimos a aquellos que aplican una misma función a cada estado del STE. Casos particulares de éstos son los operadores de prioridad ya vistos, los de encapsulamiento y los de renombrado. Vamos a demostrar que la nueva clase de operadores es compatible con ready-simulación y bisimulación, para que se vea que estamos considerando una clase razonable de operadores. Queremos ver si los resultados vistos en el capítulo anterior valen para la clase **U** y, como ya dijimos, caracterizar los operadores que son compatibles con las n-nested simulaciones.

Definición 5.1.1 (La clase U de operadores) *Sea H un conjunto de funciones parciales $f : \mathcal{P}(Act) \longrightarrow (Act \longrightarrow Act)$. Definimos el* operador $\mathcal{U}_f : \mathcal{C}_{STE} \to \mathcal{C}_{STE}$, *para todo $f \in H$, de podado y renombramiento de transiciones, como: $\mathcal{U}_f((S, i, Act, \longrightarrow)) \stackrel{\text{def}}{=} (S, i, Act, \longrightarrow^{\mathbf{U}})$, en donde $p \stackrel{b}{\longrightarrow}^{\mathbf{U}} q \stackrel{\text{def}}{\Leftrightarrow} p \stackrel{a}{\longrightarrow} q \wedge f(l(p))(a) = b$.*

Con **U** *nos referimos al conjunto de todos los operadores $\mathcal{U}_f$.*

Veamos ejemplos de como se definiren en la clase **U** algunos operadores unarios conocidos:

- El operador de renombrado ρ_h [Hoa85, Mil89, BW90], que renombra cada etiqueta del STE de acuerdo a la función de renombrado $h : Act \to Act$ se puede definir como

$\rho_h = \mathcal{U}_f$ with $f(X)(a) = h(a)$

- Los operadores de prioridad $\theta_{\geq}$ [BBK86, BW90],vistos en el capítulo anterior, que cortan las ramas de baja prioridad según $\geq$ se pueden ver como

  $\theta_{\geq} = \mathcal{U}_f$ con $f(X)(a) = a$ cuando a es maximal en X

- El operador de encapsulamiento ∂_H [Mil89, BW90], que prohíbe las transiciones cuyas etiquetas pertencen al conjunto de acciones H se define como:

  $\partial_H = \mathcal{U}_f$ with $f(X)(a) = a$ if $a \notin H$

- También podemos definir el operador identidad id, que no modifica las acciones:

  $id = \mathcal{U}_{id}$ with $id(X)(a) = a$

Una característica de éstos operadores es que se pueden componer para obtener un nuevo operador. Esto es, dados $\mathcal{U}_f, \mathcal{U}_g \in \mathbf{U}$ podemos definir $\mathcal{U}_h \in \mathbf{U}$ como $\mathcal{U}_{f\star g}(\mathcal{G})$ tal que $\mathcal{U}_h = \mathcal{U}_f \circ \mathcal{U}_g(\mathcal{G}))$. Esta propiedad de composición de los operadores nos va a servir más adelante para demostrar algunos resultados importantes.

Definición 5.1.2 (Composición en H: $\star$) *Sean* $f : \mathcal{P}(Act) \longrightarrow (Act \longrightarrow Act)$ *y* $g : \mathcal{P}(Act) \longrightarrow (Act \longrightarrow Act) \in H$. *Definimos* $f \star g : \mathcal{P}(Act) \longrightarrow (Act \longrightarrow Act)$ *como la función parcial:*
$f \star g(X)(a) = f(Y)(g(X)(a))$ *si* $g(X)(a)$ *está definido,*
donde $Y = \{g(X)(a), a \in X\}$.

Veamos entonces el resultado que nos interesa:

Teorema 5.1.3 (Composición de operadores en U) $\mathcal{U}_f \circ \mathcal{U}_g = \mathcal{U}_{f\star g}$.

Demostración: Vamos a escribir que es $\mathcal{U}_f \circ \mathcal{U}_g$ y que es $\mathcal{U}_{f\star g}$ para ver que son iguales.
Por definición de los operadores de U tenemos que
$\mathcal{U}_g(S, i, Act, \longrightarrow) = (S, i, Act, \longrightarrow^{\mathcal{U}_g})$, donde
$p \xrightarrow{b}^{\mathcal{U}_g} q$ sii $p \xrightarrow{a} q \wedge g(I(p))(a) = b$.
Ahora obtenemos $\mathcal{U}_f \circ \mathcal{U}_g$:
$\mathcal{U}_f \circ \mathcal{U}_g(S, i, Act, \longrightarrow) = (S, i, Act, \longrightarrow^{\mathcal{U}_\circ})$, donde
$p \xrightarrow{b}^{\mathcal{U}_\circ} q$ sii $p \xrightarrow{a}^{\mathcal{U}_g} q \wedge f(I_{\mathcal{U}_g}(p))(a) = b$.
Recordemos que por $I_{\mathcal{U}_g}(p)$ entendemos el conjunto de acciones iniciales del estado p en el STE $\mathcal{U}_g$. Esto es, $I_{\mathcal{U}_g}(p) = \{a : \exists q$ tal que $p \xrightarrow{a}^{\mathcal{U}_g} q\}$. Entonces $b \in I_{\mathcal{U}_g}(p))$ sii $\exists q : p \xrightarrow{a} q \wedge g(I(p))(a) = b$.
Veamos ahora que es $\mathcal{U}_{f\star g}$:
$\mathcal{U}_{f\star g}(S, i, Act, \longrightarrow) = (S, i, Act, \longrightarrow^{\star})$, donde
$p \xrightarrow{b}^{\star} q$ sii $p \xrightarrow{a} q \wedge f \star g(I(p))(a) = b$.
Por la definición 5.1.2 sabemos que $f \star g(I(p))(a) = f(Y)(g(I(p))(a))$ **con** $Y = \{g(I(p))(a), a \in I(p)\}$.
Sea $b \in Y$, entonces $\exists a \in I(p) : g(I(p))(a) = b$. Sabemos entonces por la definición de $\mathcal{U}_g$ que $\exists q : p \xrightarrow{b}^{\mathcal{U}_g} q$ y por lo tanto $b \in I_{\mathcal{U}_g}(p)$. Mas aún, todos los entonces son en realidad un sí y sólo sí, con lo cual $b \in Y$ sii $b \in I_{\mathcal{U}_g}(p)$. ■

Esta propiedad es muy interesante porque me dice, por ejemplo, que aplicar un operador de prioridad a un STE y luego aplicarle un operador de renombrado al sistema resultante da lo mismo que aplicar un único operador estático al STE.

5.2 U: el caso de ready-simulación y bisimulación

Vamos a verificar ahora que la ready-simulación y la bisimulación son congruencias para la nueva clase de operadores.

Damos primero el resultado para bisimulación: Primero vemos que la misma ready-simulación R de $\mathcal{G}_1$ en $\mathcal{G}_2$ es una ready-simulación entre $\mathcal{U}_f(\mathcal{G}_1)$ y $\mathcal{U}_f(\mathcal{G}_2)$.

Teorema 5.2.1 *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$. Para todo $\mathcal{U}_f \in \mathbf{U}$ se cumple que $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2 \Rightarrow R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_R \mathcal{U}_f(\mathcal{G}_2)$.*

Demostración: *Sea $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$.* Tomamos $p \in S_1, q \in S_2$ tales que pRq y $p \xrightarrow{b}{}^{\mathbf{U}}_1 p'$. Eso significa, por la definición de $\mathcal{U}_f$, que $\exists a \in Act : p \xrightarrow{a}_1 p'$ y $f(I(p))(a) = b$. Como R es una simulación sabemos que $\exists q' \in S_2 : q \xrightarrow{a}_2 q'$ y $p'Rq'$. Pero además $I(p) = I(q)$, porque R es una ready-simulación, entonces $f(I(q))(a) = f(I(p))(a) = b$. Por lo tanto $q \xrightarrow{b}{}^{\mathbf{U}}_2 q'$ y $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq \mathcal{U}_f(\mathcal{G}_2)$.

Veamos que los conjuntos de acciones iniciales de p' y q' son iguales: Sea $b \in I_{\mathbf{U}}(p)$ entonces $p \xrightarrow{b}{}^{\mathbf{U}}_1 p'$. Como R es una simulación $q \xrightarrow{b}{}^{\mathbf{U}}_2 q' \Rightarrow b \in I_{\mathbf{U}}(q)$ y $I_{\mathbf{U}}(p) \subseteq I_{\mathbf{U}}(q)$. Ahora sea $b \in I_{\mathbf{U}}(q)$. Por la definición de $\mathcal{U}_f$ sabemos que $\exists a \in Act : q \xrightarrow{a}_2 q'$ y $f(I(q))(a) = b$. Pero $a \in I(q)$ y $I(q) = I(p)$ tenemos que $a \in I(p)$. Además $f(I(p))(a) = f(I(q))(a) = b$ y finalmente $b \in I_{\mathbf{U}}(p)$. Así $I_{\mathbf{U}}(q) \subseteq I_{\mathbf{U}}(p)$ y obtenemos que $I_{\mathbf{U}}(p) = I_{\mathbf{U}}(q)$.

Finalmente $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_R \mathcal{U}_f(\mathcal{G}_2)$. ■

Corolario 5.2.2 (Ready-simulación es una congruencia para U) *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$. Para todo $\mathcal{U}_f \in \mathbf{U}$ se cumple que $\mathcal{G}_1 \rightleftharpoons_R \mathcal{G}_2 \Rightarrow \mathcal{U}_f(\mathcal{G}_1) \rightleftharpoons_R \mathcal{U}_f(\mathcal{G}_2)$.*

Para demostrar que la bisimulación también es una congruencia para la clase **U** vamos a tener en cuenta que toda bisimulación R es una ready-simulación:

Lema 5.2.3 *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$ tales que $R : \mathcal{G}_1 \leftrightarrows \mathcal{G}_2$. Entonces $R, R^{-1} : \mathcal{G}_1 \rightleftharpoons_R \mathcal{G}_2$.*

Demostración: Sea $R : \mathcal{G}_1 \leftrightarrows \mathcal{G}_2$. Por definición de bisimulación sabemos que $R : \mathcal{G}_1 \subseteq \mathcal{G}_2$ y $R^{-1} : \mathcal{G}_2 \subseteq \mathcal{G}_1$. Sean $p \in S_1, q \in S_2$ tales que $pRq \wedge qR^{-1}p$. Supongo que $I(p) \neq I(q)$, entonces $\exists a \in Act : a \in I(p)$ y $a \notin I(q)$ (es lo mismo si $a \in I(q)$ y $a \notin I(p)$). Entonces $p \xrightarrow{a} p' \wedge q \not\xrightarrow{a}$, lo cual es absurdo porque R es una bisimulación. Por lo tanto $R, R^{-1} : \mathcal{G}_1 \rightleftharpoons_R \mathcal{G}_2$. ■

Demostramos ahora que la bisimulación es una congruencia para los operadores de la clase **U**.

Teorema 5.2.4 (Bisimulación es una congruencia para U) *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$. Para todo $\mathcal{U}_f \in \mathbf{U}$ se cumple que $\mathcal{G}_1 \leftrightarrows \mathcal{G}_2 \Rightarrow \mathcal{U}_f(\mathcal{G}_1) \leftrightarrows \mathcal{U}_f(\mathcal{G}_2)$.*

Demostración: Probamos algo más fuerte aún porque decimos que la misma bisimulación R de $\mathcal{G}_1$ en $\mathcal{G}_2$ resulta ser bisimulación entre $\mathcal{U}_f(\mathcal{G}_1)$ y $\mathcal{U}_f(\mathcal{G}_2)$.

Recordemos que una bisimulación R es una relación tal que ella y su inversa (R^{-1}) son simulaciones. Sea $R : \mathcal{G}_1 \leftrightarrows \mathcal{G}_2$. Vimos en el lema 5.2.3 que $R, R^{-1} : \mathcal{G}_1 \rightleftharpoons_R \mathcal{G}_2$ y, por el teorema 5.2.1 y su corolario 5.2.2, $R, R^{-1} : \mathcal{U}_f(\mathcal{G}_1) \rightleftharpoons_R \mathcal{U}_f(\mathcal{G}_2)$. Entonces $R : \mathcal{U}_f(\mathcal{G}_1) \subsetneq \mathcal{U}_f(\mathcal{G}_2) \wedge R^{-1} : \mathcal{U}_f(\mathcal{G}_2) \subsetneq \mathcal{U}_f(\mathcal{G}_1)$.

Finalmente $R : \mathcal{U}_f(\mathcal{G}_1) \leftrightarrows \mathcal{U}_f(\mathcal{G}_2)$. ■

Ya que consideramos que la clase de operadores que definimos es lo suficientemente razonable nos internamos en el tema que nos interesa. Analizamos el comportamiento de las n-nested simulaciones con respecto a esta clase.

5.3 U y la n-nested simulación

Tal como era de esperarse la clase U de operadores no es compatible con las n-nested simulaciones.

Teorema 5.3.1 ($\rightleftharpoons_n$ no es una congruencia para la clase U) *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$. Existe $\mathcal{U}_f \in$ U tal que $\mathcal{G}_1 \rightleftharpoons_n \mathcal{G}_2$ y $\mathcal{U}_f(\mathcal{G}_1) \not\rightleftharpoons_n \mathcal{U}_f(\mathcal{G}_2)$.*

Este resultado es inmediato si tenemos en cuenta que los operadores de prioridad son casos particulares de $\mathcal{U}_f$ y demostramos en el teorema 4.1.4 que esa clase de operadores no es compatible con las n-nested simulaciones.

Buscamos ahora el subconjunto de operadores que son compatibles con las n-nested simulaciones. Para ello definimos los operadores monótonos dentro de la clase U.

Definición 5.3.2 *Un operador $\mathcal{U}_f \in$ U es monótono sii la función f es monótona.*

Definición 5.3.3 (Función monótona) *Una función parcial $f : \mathcal{P}(Act) \longrightarrow (Act \longrightarrow Act)$ es una* función monótona *sii $\forall a, b \in Act, X, Y \subseteq Act : f(X)(a) = b$ implica que $f(X \cup Y)(a) = b$.*

Los operadores monótonos forman un subconjunto de la clase U. A continuación vamos a demostrar que las n-nested simulaciones son congruencias para los $\mathcal{U}_f$ solamente cuando $\mathcal{U}_f$ sea monótono. Más aún, vamos a probar que todos los demás operadores de U no son compatibles con las n-nested simulaciones. Para ello necesitamos una serie de resultados previos.

Lema 5.3.4 *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$ y $p \in S_1, q \in S_2$ tales que $R : \mathcal{G}_1 \subsetneq \mathcal{G}_2 \wedge pRq$. Entonces $I(p) \subseteq I(q)$.*

Demostración: Sea $R : \mathcal{G}_1 \subsetneq \mathcal{G}_2$ y pRq. Tomamos $a \in I(p)$ entonces $\exists p' \in S_1 : p \xrightarrow{a}_1 p'$. Por definición de simulación sabemos que $\exists q' \in S_2 : q \xrightarrow{a}_2 q'$. Por lo tanto $a \in I(q)$. ■

A continuación vamos a dar pares de STE tales que no son n-nested similares. Luego vamos a utilizar éstos sistemas para nuestra demostración.

Nota 5.3.5 *Para mayor claridad en las pruebas vamos a usar X como la suma no determinística de $x_1, x_2, \ldots, x_n$. Es decir X equivale a $x_1 + x_2 + \ldots + x_n$.*

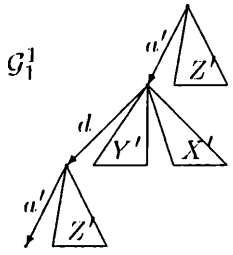
Teorema 5.3.6 *Sean $\mathcal{G}_1^n, \mathcal{G}_2^n \in \mathcal{C}_{STE}$ los STE dados a partir de los términos u'_n y v'_n:*

$$\begin{array}{ll} u'_0 = c(a' + Z') + Y'' & v'_0 = d(a' + Z') + Y' + X' \\ u'_{n+1} = a'.v'_n + Z' & v'_{n+1} = a'.u'_n + Z' + a'.v'_n \end{array}$$

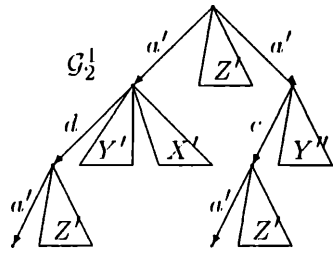
Entonces $\mathcal{G}_1^n \not\rightleftharpoons_n \mathcal{G}_2^n$.

Demostración: Para demostrar que dos STE no son n-nested similares damos una función HML que se cumpla para solamente uno de ellos. Sea $h_n \in \mathcal{L}n$ la función HML dada por $h_0 = \langle a' \rangle \langle c \rangle \langle a' \rangle T$, $h_{n+1} = \langle a' \rangle \neg h_n$. Veamos que $\mathcal{G}_1^n \not\models h_{n-1} \wedge \mathcal{G}_2^n \models h_{n-1}$.

(**caso** $n = 1$) Gráficamente los STE para el caso $n = 1$ corresponden a:



$a'(d(a' + Z') + Y' + X') + Z'$



$\mathcal{G}_1^1 + a'(c(a' + Z') + Y'')$

Tenemos $\mathcal{G}_1^1 = a'(d(a'+Z')+Y'+X')+Z'$, $\mathcal{G}_2^1 = \mathcal{G}_1^1 + a'(c(a'+Z')+Y'')$ y $h_0 = \langle a' \rangle \langle c \rangle \langle a' \rangle T$.

Ahora, $\mathcal{G}_1^1 \not\models h_0 \Leftrightarrow a'(d(a' + Z') + Y' + X') + Z' \not\models \langle a' \rangle \langle c \rangle \langle a' \rangle T \Leftrightarrow d(a' + Z') + Y' + X' \not\models \langle c \rangle \langle a' \rangle T$. Lo cual se ve que es cierto.

Para $\mathcal{G}_2^1$: $\mathcal{G}_2^1 \models h_0 \Leftrightarrow \mathcal{G}_1^1 + a'(c(a' + Z') + Y'') \models \langle a' \rangle \langle c \rangle \langle a' \rangle T \Leftrightarrow a'(c(a' + Z') + Y'') \models \langle a' \rangle \langle c \rangle \langle a' \rangle T \Leftrightarrow c(a' + Z') + Y'' \models \langle c \rangle \langle a' \rangle T$, que vemos que es cierto.

(**caso** $n \geq 1$) $\mathcal{G}_1^{n+1} \not\models h_n \Leftrightarrow a'\mathcal{G}_2^n + Z' \not\models \langle a' \rangle \neg h_{n-1} \Leftrightarrow \mathcal{G}_2^n \not\models \neg h_{n-1} \Leftrightarrow \mathcal{G}_2^n \models h_{n-1}$, que es cierto por hipótesis inductiva.

Finally $\mathcal{G}_2^{n+1} \models h_n \Leftrightarrow a'\mathcal{G}_1^n + Z' + a'\mathcal{G}_2^n \models \langle a' \rangle \neg h_{n-1} \Leftrightarrow a'\mathcal{G}_1^n \models \langle a' \rangle \neg h_{n-1} \Leftrightarrow \mathcal{G}_1^n \models \neg h_{n-1} \Leftrightarrow \mathcal{G}_1^n \not\models h_{n-1}$, lo cual es verdadero por hipótesis. ■

Aqui está entonces una de las contribuciones más importantes de este trabajo. Demostramos que los operadores de U para los cuales $\rightleftharpoons_n$ es una congruencia son solamente los monótonos.

Teorema 5.3.7 ($\rightleftharpoons_n$ como congruencia) *$\rightleftharpoons_n$ es una congruencia para $\mathcal{U}_f \in \mathbf{U}$ sii f es monótono.*

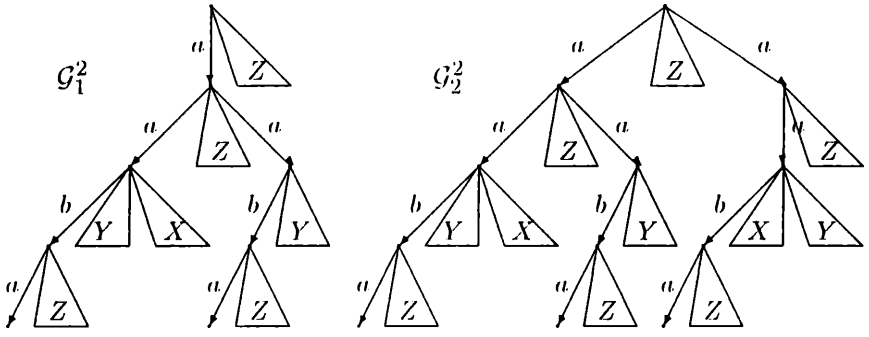
Demostración: ($\Rightarrow$) Para cada f no monótona damos un par de STE n-nested similares tales que al aplicarles el operador $\mathcal{U}_f$ obtengo STE que no son n-nested similares. Esto es, siempre que tengo un operador no monótono puedo ver que las n-nested simulaciones no son congruencias.

Sea f una función no monótona. Entonces $\exists b, c \in Act, X, Y \in \mathcal{P}(Act)$ tales que $f(Y)(b) = c$ y $f(X \cup Y)(b) \neq c$. Como f es una función parcial es posible que $f(X \cup Y)(b)$ no esté definida o que sea igual a $d \neq c$. No perdemos generalidad si consideramos que $f(X \cup Y)(b) = d$. Además podemos encontrar $a \in Act, Z \in \mathcal{P}(Act)$ tales que $f(Z)(a) = a'$.

Definimos ahora los STE $\mathcal{G}_1^n, \mathcal{G}_2^n \in \mathcal{C}_{STE}$ a partir de los términos u_n y v_n respectivamente. Esos STE son n-nested similares por como los construimos a partir de los STE del ejemplo 2.3.13.

$$u_0 = b(a + Z) + Y \qquad v_0 = b(a + Z) + Y + X$$
$$u_{n+1} = a.v_n + Z \qquad v_{n+1} = a.u_n + Z + a.v_n$$

Para el caso en que $n = 2$ obtenemos los STE de la figura tales que $R_1, Q_1 : \mathcal{G}_1^2 \rightleftharpoons_2 \mathcal{G}_2^2$.



$$R_1 = Id \cup \{(a(a(b(a + Z) + X + Y) + Z + a(b(a + Z) + Y)) + Z,$$
$$a(a(b(a + Z) + X + Y) + Z + a(b(a + Z) + Y)) + Z + a(a(b(a + Z) + X + Y) + Z))\}$$

$$R_2 = R_1^{-1} \cup \{(a(a(b(a + Z) + X + Y) + Z), a(b(a + Z) + X + Y) + Z + a(b(a + Z) + Y))\}$$

$$Q_1 = R_2$$

$$Q_2 = Q_1^{-1} \cup \{(b(a + Z) + Y, b(a + Z) + X + Y)\}$$

Tenemos entonces pares de STE n-nested equivalentes ($\mathcal{G}_1^n \rightleftharpoons_n \mathcal{G}_2^n$). Aplicamos el operador $\mathcal{U}_f$ en ambos y queremos ver que $\mathcal{U}_f(\mathcal{G}_1^n) \not\rightleftharpoons_n \mathcal{U}_f(\mathcal{G}_2^n)$. Para poder escribir la suponesmos que la definición de f se completa de la siguiente manera: $f(X \cup Y \cup \{b\})(x_i) = x_i'$, $f(X \cup Y \cup \{b\})(y_i) = y_i'$, $f(Y \cup \{b\})(y_i) = y_i'' \wedge f(Z \cup a)(z_i) = z_i'$. Los STE $\mathcal{U}_f(\mathcal{G}_1^n)$ y $\mathcal{U}_f(\mathcal{G}_2^n)$ obtenidos coinciden con los STE del teorema 5.3.6. Ya vimos que esos STE no son n-nested similares y por lo tanto $\rightleftharpoons_n$ no es una congruencia.

($\Leftarrow$) Demostramos ahora que cuando $\mathcal{U}_f$ es un operador monótono las n-nested simulaciones son congruencias para él.

Dada $f : \mathcal{P}(Act) \to (Act \cdot Act)$, una función monótona veamos que $R : \mathcal{G}_1^n \subseteq_n \mathcal{G}_2^n$ implica $R : \mathcal{U}_f(\mathcal{G}_1^n) \subseteq_n \mathcal{U}_f(\mathcal{G}_2^n)$.

En el caso en que $n = 1$: Sea $R : \mathcal{G}_1^1 \subseteq_1 \mathcal{G}_2^1$, con pRq y $p \xrightarrow{b}_1 p'$. Por la definición de $\mathcal{U}_f$ sabemos que $\exists a \in Act : f(I(p))(a) = b$. Como R es una simulación se cumple que

$\exists q' \in S_2 : q \xrightarrow{a}_2 q' \wedge p' R q'$. Como f es monótona y $I(p) \subseteq I(q)$ (por el lema 5.3.4), sabemos que $f(I(q))(a) = b$. Por lo tanto $q \xrightarrow[b]{\mathbf{U}}_2 q'$ y $R : \mathcal{U}_f(\mathcal{G}_1^1) \subseteq \mathcal{U}_f(\mathcal{G}_2^1)$

Cuando $n > 1$: $R : \mathcal{G}_1^n \subseteq_n \mathcal{G}_2^n$. Por la definición de n-nested simulación $R : \mathcal{G}_1^n \subseteq \mathcal{G}_2^n$ y, a partir del caso $n = 1$, sabemos que $R : \mathcal{U}_f(\mathcal{G}_1^n) \subseteq \mathcal{U}_f(\mathcal{G}_2^n)$. Además sabemos que existe otra relación Q tal que $Q : \mathcal{G}_2^n \subseteq_{n-1} \mathcal{G}_1^n$, con $R^{-1} \subseteq Q$. Entonces, por hipótesis inductiva, $Q : \mathcal{U}_f(\mathcal{G}_2^n) \subseteq_{n-1} \mathcal{U}_f(\mathcal{G}_1^n)$. Finalmente $R : \mathcal{U}_f(\mathcal{G}_1^n) \subseteq_n \mathcal{U}_f(\mathcal{G}_2^n)$, con lo que se demuestra que si $\mathcal{U}_f$ es monótona, luego $\subseteq_n$ es una precongruencia. Surge inmediatamente que $\rightleftharpoons_n$ es una congruencia. ■

De éste teorema podemos obtener también la caracterización de los operadores compatibles con la simulación, teniendo en cuenta que $\rightleftharpoons$ es equivalente a $\rightleftharpoons_1$.

Corolario 5.3.8 (Simulación como congruencia) *$\rightleftharpoons$ es una congruencia para $\mathcal{U}_f \in \mathbf{U}$ sii $\mathcal{U}_f$ es monótono, i.e. $\forall a, b \in Act, X, Y \subseteq Act : f(X)(a) = b$ implica $f(X \cup Y)(a) = b$.*

A pesar de que la n-nested simulación no es una congruencia para los operadores de la clase **U** podemos "acotar su comportamiento", como hicimos en la sección 4.2. Esto es, dados dos STE n-nested similares $\mathcal{G}_1$ y $\mathcal{G}_2$ y un operador $\mathcal{U}_f$ de **U** podemos restringir las clases de equivalencia a las que pertenecerán $\mathcal{U}_f(\mathcal{G}_1)$ y $\mathcal{U}_f(\mathcal{G}_2)$. Más aún, vamos a demostrar que en esas condiciones $\mathcal{U}_f(\mathcal{G}_1) \rightleftharpoons_{n-1} \mathcal{U}_f(\mathcal{G}_2)$. Para ello necesitamos una serie de resultados previos.

Lema 5.3.9 *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$. $R : \mathcal{G}_1 \subseteq_n \mathcal{G}_2, n \geq 2 \Rightarrow R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_R \mathcal{U}_f(\mathcal{G}_2)$ y además $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq \mathcal{U}_f(\mathcal{G}_2)$.*

Demostración: Si $R : \mathcal{G}_1 \subseteq_n \mathcal{G}_2$ entonces, por el lema 4.1.6, tenemos $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$. Ademas, por el teorema 5.2.1, $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_R \mathcal{U}_f(\mathcal{G}_2)$. $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq \mathcal{U}_f(\mathcal{G}_2)$ se obtiene inmediatamente de la definición de ready simulación, ya que toda ready-simulación es una simulación. ■

Volvamos por un momento al teorema 4.1.8 y veamos que lo que se cumplía para los operadores de prioridad se puede extender a toda la clase **U**. Esta nueva demostración sigue los mismos lineamientos que la anterior.

Teorema 5.3.10 *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$, tales que $R : \mathcal{G}_1 \subseteq_{n+1} \mathcal{G}_2, n \geq 1$. Entonces, para todo operador $\mathcal{U}_f \in \mathbf{U}$, se cumple que $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_n \mathcal{U}_f(\mathcal{G}_2)$.*

Demostración: Nuevamente probamos que la misma relación R es la que hace n-nested similares a $\mathcal{U}_f(\mathcal{G}_1)$ y $\mathcal{U}_f(\mathcal{G}_2)$.

(caso $n = 1$) A partir del lema 5.3.9 sabemos que $R : \mathcal{G}_1 \subseteq_2 \mathcal{G}_2$ implica $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq \mathcal{U}_f(\mathcal{G}_2)$, que coincide $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_1 \mathcal{U}_f(\mathcal{G}_2)$.

(caso $n > 1$) Sea $R : \mathcal{G}_1 \subseteq_{n+1} \mathcal{G}_2$. Queremos ver que $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_n \mathcal{U}_f(\mathcal{G}_2)$. Para ello usamos la definición de n-nested simulación y probamos que $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq \mathcal{U}_f(\mathcal{G}_2)$ y que $\exists Q : \mathcal{G}_2 \subseteq_{n-1} \mathcal{G}_1$, con $R^{-1} \subseteq Q$.

A partir del lema 5.3.9 sabemos que $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq \mathcal{U}_f(\mathcal{G}_2)$. Además, por definición de n-nested simulación tenemos que $\exists Q : \mathcal{G}_2 \subseteq_n \mathcal{G}_1 \wedge R^{-1} \subseteq Q$ y que, por hipótesis inductiva, $Q : \mathcal{U}_f(\mathcal{G}_2) \subseteq_{n-1} \mathcal{U}_f(\mathcal{G}_1)$. Por lo tanto $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_n \mathcal{U}_f(\mathcal{G}_2)$. ■

Corolario 5.3.11 (Comportamiento de U con respecto a la n-nested simulación) *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$, tales que $\mathcal{G}_1 \rightleftharpoons_{n+1} \mathcal{G}_2, n \geq 1$. Entonces, para todo operador $\mathcal{U}_f$, se cumple que $\mathcal{U}_f(\mathcal{G}_1) \rightleftharpoons_n \mathcal{U}_f(\mathcal{G}_2)$.*

Siguiendo el mismo razonamiento que en el capítulo anterior nos preguntamos por las equivalencias incluidas en $\rightleftharpoons_n$ que son congruencias para la clase **U**. ¿Cuál es la mayor congruencia para los $\mathcal{U}_f$ contenida en la n-nested simulación? ¿ Coincide con la n-nested congruencia definida para los operadores de prioridad? De no ser así, ¿ coincide en este caso con la bisimulación?

Definición 5.3.12 (U congruencia) *Definimos la* **U** congruencia, *notación $\sim_n^{\mathbf{U}}$, como la congruencia, con respecto a la clase* **U**, *más gruesa contenida en la n-nested simulación.*

A continuación damos una definición alternativa de **U** congruencia, con el único fin de facilitar las demostraciones.

Lema 5.3.13 (Definición alternativa de $\sim_n^{\mathbf{U}}$) *$\mathcal{G}_1 \sim_n^{\mathbf{U}} \mathcal{G}_2$ sii para todo $n \geq 1$ se cumple que $\forall f : \mathcal{U}_f(\mathcal{G}_1) \rightleftharpoons_n \mathcal{U}_f(\mathcal{G}_2)$.*

Demostración: Usamos como notación: $\mathcal{G}_1 \equiv_n \mathcal{G}_2$ sii $\forall f : \mathcal{U}_f(\mathcal{G}_1) \rightleftharpoons_n \mathcal{U}_f(\mathcal{G}_2)$. **(caso $\subseteq$)** Ya que $\sim_n^{\mathbf{U}}$ es la mayor congruencia incluida en $\rightleftharpoons_n$, es suficiente con demostrar que $\equiv_n \subseteq \rightleftharpoons_n$ y que $\equiv_n$ es una congruencia para toda $\mathcal{U}_f$. Entonces, tenemos que: $\mathcal{G}_1 \equiv_n \mathcal{G}_2$ significa que $\forall f : \mathcal{U}_f(\mathcal{G}_1) \rightleftharpoons_n \mathcal{U}_f(\mathcal{G}_2)$. En particular, vale para $f = id$ y entonces $\mathcal{U}_{id}(\mathcal{G}_1) \rightleftharpoons_n \mathcal{U}_{id}(\mathcal{G}_2) \Rightarrow \mathcal{G}_1 \rightleftharpoons_n \mathcal{G}_2$, es decir, $\equiv_n \subseteq \rightleftharpoons_n$.

Ademas,

$$
\begin{aligned}
\mathcal{G}_1 \equiv_n \mathcal{G}_2 \quad & \Rightarrow \forall f : \mathcal{U}_f(\mathcal{G}_1) \rightleftharpoons_n \mathcal{U}_f(\mathcal{G}_2) \\
& \text{en particular es cierto para } \mathcal{U}_{h*g} \\
& \Rightarrow \forall g \forall h : \mathcal{U}_{h*g}(\mathcal{G}_1) \rightleftharpoons_n \mathcal{U}_{h*g}(\mathcal{G}_2) \\
& \Rightarrow \forall g \forall h : \mathcal{U}_h(\mathcal{U}_g(\mathcal{G}_1)) \rightleftharpoons_n \mathcal{U}_h(\mathcal{U}_g(\mathcal{G}_2)) \qquad \text{(por el teorema 5.1.3)} \\
& \Rightarrow \forall g : \mathcal{U}_g(\mathcal{G}_1) \equiv_n \mathcal{U}_g(\mathcal{G}_2)
\end{aligned}
$$

lo que muestra que $\equiv_n$ es una congruencia.

(caso $\supseteq$) Ahora debemos mostrar que $\mathcal{G}_1 \sim_n^{\mathbf{U}} \mathcal{G}_2 \Rightarrow \mathcal{G}_1 \equiv_n \mathcal{G}_2$, o lo que es lo mismo: $\mathcal{G}_1 \not\equiv_n \mathcal{G}_2 \Rightarrow \mathcal{G}_1 \not\sim_n^{\mathbf{U}} \mathcal{G}_2$.

Supongamos $\mathcal{G}_1 \not\equiv_n \mathcal{G}_2$. Luego, por definición de $\equiv_n$, $\exists f : \mathcal{U}_f(\mathcal{G}_1) \not\rightleftharpoons_n \mathcal{U}_f(\mathcal{G}_2)$. Dado que $\sim_n^{\mathbf{U}} \subseteq \rightleftharpoons_n$, esto implica que $\mathcal{U}_f(\mathcal{G}_1) \not\sim_n^{\mathbf{U}} \mathcal{U}_f(\mathcal{G}_2)$. Finalmente $\mathcal{G}_1 \not\sim_n^{\mathbf{U}} \mathcal{G}_2$ porque sabemos que $\sim_n^{\mathbf{U}}$ es una congruencia para los operadores de **U**. ■

Ubicamos ahora la congruencia recientemente definida en el reticulado de las equivalencias semánticas conocidas. Vemos que no coincide con la bisimulación sino que las distintas $\sim_n^{\mathbf{U}}$ se encuentran intercaladas entre las n-nested simulaciones.

Teorema 5.3.14 (U congruencia en el reticulado de la equivalencias semánticas) *Para todo n, $\rightleftharpoons_{n+1} \subset \sim_n^{\mathbf{U}} \subset \rightleftharpoons_n$.*

Demostración: De la definición anterior sabemos que $\sim_n^{\mathbf{U}} \subseteq \rightleftharpoons_n$ y, como $\rightleftharpoons_n$ no es una congruencia para **U** (ver teorema 5.3.1), vemos que $\sim_n^{\mathbf{U}} \neq \rightleftharpoons_n$. Por lo tanto $\sim_n^{\mathbf{U}} \subset \rightleftharpoons_n$.

Sabemos que $\rightleftharpoons_{n+1} \neq \sim_n^{\mathbf{U}}$ porque en el teorema 4.1.10 tenemos pares de STE $\mathcal{G}_1^n$ y $\mathcal{G}_2^n$ tales que $\mathcal{G}_1^n \sim_n^{\mathbf{U}} \mathcal{G}_2^n$ y $\mathcal{G}_1^n \not\rightleftharpoons_{n+1} \mathcal{G}_2^n$. Falta ver que $\rightleftharpoons_{n+1} \subseteq \sim_n^{\mathbf{U}}$.

Supongamos que $\rightleftharpoons_{n+1} \not\subseteq \sim_n^{\mathbf{U}}$. Entonces existe una secuencia de operadores $\mathcal{U}_{f_1}, \ldots, \mathcal{U}_{f_j} \in \mathbf{U}$ tal que $\mathcal{G}_1 \rightleftharpoons_{n+1} \mathcal{G}_2$ y $\mathcal{U}_{f_1} \circ \ldots \circ \mathcal{U}_{f_j}(\mathcal{G}_1) \not\rightleftharpoons_n \mathcal{U}_{f_1} \circ \ldots \circ \mathcal{U}_{f_j}(\mathcal{G}_2)$. Pero vimos en el teorema 5.1.3 que estos operadores se pueden componer, con lo que existe un $\mathcal{U}_f \in \mathbf{U}$ tal que $\mathcal{U}_f = \mathcal{U}_{f_1} \circ \ldots \circ \mathcal{U}_{f_j}$. Entonces tenemos que $\mathcal{U}_f(\mathcal{G}_1) \not\rightleftharpoons_n \mathcal{U}_f(\mathcal{G}_2)$, lo cual es un absurdo por el teorema 5.3.11. Finalmente $\rightleftharpoons_{n+1} \subseteq \sim_n^{\mathbf{U}}$. ■

Corolario 5.3.15 (n-nested congruencia en el reticulado de equivalencias semánticas) $\rightleftharpoons_{n+1} \subset \sim_n^{\mathbf{U}} \subseteq \sim_n^{\theta} \subset \rightleftharpoons_n$.

Demostración: Vimos al final del capítulo 4 que $\sim_n^{\theta} \subset \rightleftharpoons_n$. Además, que $\sim_n^{\mathbf{U}}$ sea una congruencia para todos los operadores de U, significa que en particular lo es para los operadores de prioridad, entonces: $\sim_n^{\mathbf{U}} \subseteq \sim_n^{\theta}$. ■

Lo que se hizo en éste capítulo fue detectar que la n-nested simulación no es una congruencia para los operadores de la clase U. A partir de ahí se caracterizaron los operadores que presentaban problemas y, paralelamente, se buscó una equivalencia alternativa para usar en sistemas en que esos operadores tengan sentido. Se definió la $\sim_n^{\mathbf{U}}$, más fina que la $\rightleftharpoons_{n+1}$ y más gruesa que la $\rightleftharpoons_n$.

Capítulo 6

La n-nested ready simulación

Ya vimos que $\rightleftharpoons_n$ no es congruencia para U. Encontramos una familia infinita de equivalencias que sí lo son, las U congruencias, pero no pudimos encontrar su caracterización. Seguimos interesados en en encontrar una familia infinita de equivalencias entre la ready-s imulación y la bisimulación con buenas propiedades, para usar en los casos en que los operadores de U tengan sentido. Esa búsqueda nos llevó a una familia de equivalencias semánticas, las cuales pudimos caracterizar relacionalmente y que tienen buen comportamiento con respecto a los operadores $\mathcal{U}_f$. Dimos en llamar n-nested ready simulaciones a esas nuevas equivalencias, ya que incluyen en las n-nested simulaciones la noción de igualdad para los conjuntos de acciones iniciales, tal como se vio para la ready-simulación.

6.1 La equivalencia

Definición 6.1.1 (n-nested ready simulación) *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$. Definimos la* n-nested ready simulación *$R \subseteq S_1 \times S_2$ de $\mathcal{G}_1$ en $\mathcal{G}_2$, notación $R : \mathcal{G}_1 \subseteq_{R,n} \mathcal{G}_2$, como la relación que satisface:*

$$R : \mathcal{G}_1 \subseteq_{R,0} \mathcal{G}_2 \text{ sii } i_1 R i_2 \text{ y } pRq \text{ implica } I(p) = I(q)$$

$$R : \mathcal{G}_1 \subseteq_{R,n+1} \mathcal{G}_2 \text{ sii } R : \mathcal{G}_1 \subseteq \mathcal{G}_2 \wedge \exists Q : \mathcal{G}_2 \subseteq_{R,n} \mathcal{G}_1 \text{ con } R^{-1} \subseteq Q$$

$\mathcal{G}_1$ es n-nested ready simulado *por $\mathcal{G}_2$, notación $\mathcal{G}_1 \subseteq_{R,n} \mathcal{G}_2$, si existe $R : \mathcal{G}_1 \subseteq_{R,n} \mathcal{G}_2$. $\mathcal{G}_1$ y $\mathcal{G}_2$ son* n-nested ready similares, *notación $\mathcal{G}_1 \rightleftharpoons_{R,n} \mathcal{G}_2$, si $\mathcal{G}_1 \subseteq_{R,n} \mathcal{G}_2$ y $\mathcal{G}_2 \subseteq_{R,n} \mathcal{G}_1$.*

6.2 Propiedades de la n-nested ready simulación

Veamos que esa nueva familia de equivalencias semánticas es una congruencia para la clase de operadores definida en el capítulo anterior.

Teorema 6.2.1 *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$ y $\mathcal{U}_f$ un operador de la clase* U. *Para todo n, $R : \mathcal{G}_1 \subseteq_{R,n} \mathcal{G}_2 \Rightarrow R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_{R,n} \mathcal{U}_f(\mathcal{G}_2)$.*

Demostración: (**caso** $n = 1$) Sea $R : \mathcal{G}_1 \subseteq_{R,1} \mathcal{G}_2$. Entonces, por definición de $\rightleftharpoons_{R,n}$, sabemos que $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$. A partir del teorema 5.2.1 sabemos $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_R \mathcal{U}_f(\mathcal{G}_2)$. Además $i_1 R i_2$ y así $\mathcal{U}_f(\mathcal{G}_1) \subseteq_{R,1} \mathcal{U}_f(\mathcal{G}_2)$.

(caso $n > 1$) Sea $R : \mathcal{G}_1 \subseteq_{R,n} \mathcal{G}_2$. Entonces $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$ y $\exists Q : \mathcal{G}_2 \subseteq_{R,n-1} \mathcal{G}_1, R^{-1} \subseteq Q$. Nuevamente a partir del teorema 5.2.1 obtenemos que $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_R \mathcal{U}_f(\mathcal{G}_2)$ y por hipótesis inductiva $Q : \mathcal{U}_f(\mathcal{G}_2) \subseteq_{R,n-1} \mathcal{U}_f(\mathcal{G}_1)$. Finalmente $R_1 : \mathcal{U}_f(\mathcal{G}_1) \subseteq_{R,n} \mathcal{U}_f(\mathcal{G}_2)$. ■

Corolario 6.2.2 ($\rightleftharpoons_{R,n}$ es una congruencia para U) *Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}, \mathcal{U}_f \in \mathbf{U}$. Para todo n, $\mathcal{G}_1 \rightleftharpoons_{R,n} \mathcal{G}_2 \Rightarrow \mathcal{U}_f(\mathcal{G}_1) \rightleftharpoons_{R,n} \mathcal{U}_f(\mathcal{G}_2)$.*

Vemos en el corolario anterior que esta nueva equivalencia tiene el comportamiento deseado. Es posible que $\rightleftharpoons_{R,n}$ en realidad coincida con la U congruencia. Verifiquemos para ello la ubicación de las nuevas equivalencias en el reticulado de las equivalencias semánticas, es importante saber si esta familia infinita se intercala entre las n-nested simulaciones.

Teorema 6.2.3 $R : \mathcal{G}_1 \subseteq_{n+1} \mathcal{G}_2 \Rightarrow R : \mathcal{G}_1 \subseteq_{R,n} \mathcal{G}_2$.

Demostración: (caso $n = 1$) Sea $R : \mathcal{G}_1 \subseteq_2 \mathcal{G}_2$. Entonces $R : \mathcal{G}_1 \subseteq \mathcal{G}_2$ y $\exists Q : \mathcal{G}_2 \subseteq_1 \mathcal{G}_1, R^{-1} \subseteq Q$. Supongamos que pRq, con $I(p) \neq I(q)$. Si $a \in I(p) - I(q) \Rightarrow p \xrightarrow{a}_1 p'$ y $q \not\xrightarrow{a}_2$. Esto es absurdo porque R_1 es una simulación. Si $a \in I(q) - I(p) \Rightarrow q \xrightarrow{a}_2 q'$ y $p \not\xrightarrow{a}_1$. Pero pRq, entonces $qQp \Rightarrow p \xrightarrow{a}_1 p'$. Absurdo.

Por lo tanto $I(q) = I(p)$ y $\mathcal{G}_1 \subseteq_{R,1} \mathcal{G}_2$.

(caso $n > 1$) Sea $R : \mathcal{G}_1 \subseteq_{n+2} \mathcal{G}_2$. Entonces $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$ y $\exists Q : \mathcal{G}_2 \subseteq_{n+1} \mathcal{G}_1, R_1^{-1} \subseteq Q$. Por hipótesis inductiva $Q : \mathcal{G}_2 \subseteq_{R,n} \mathcal{G}_1$. Finalmente $R_1 : \mathcal{G}_1 \subseteq_{R,n+1} \mathcal{G}_2$. ■

Corolario 6.2.4 $\rightleftharpoons_{n+1} \subseteq \rightleftharpoons_{R,n}$.

Veamos ahora que la nueva equivalencia no coincide con la U congruencia sino que es una equivalencia más fina (ver el reticulado de las equivalencias semánticas, figura 6.1).

Teorema 6.2.5 (n-nested ready simulación en el reticulado de equivalencias semánticas) $\rightleftharpoons_{n+1} \subset \rightleftharpoons_{R,n} \subset \sim_n^{\mathbf{U}}$

Demostración:

Sean $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{STE}$ tales que $\mathcal{G}_1 \rightleftharpoons_{R,n} \mathcal{G}_2$. Ya demostramos que $\rightleftharpoons_{R,n}$ es una congruencia para los operadores $\mathcal{U}_f$. Por lo tanto $\mathcal{G}_1 \sim_n^{\mathbf{U}} \mathcal{G}_2$.

Veamos que $\rightleftharpoons_{R,n} \neq \sim_n^{\mathbf{U}}$: consideremos los STE $\mathcal{G}_1$ y $\mathcal{G}_2$ de la figura 6.2. Por haber sido construídos según las pautas del ejemplo 2.3.13 sabemos que son 2-nested similares. Probamos que efectivamente lo son usando las relaciones: $R_1 = Id \cup \{(a(a(aab+a)+aa(a+ab)), a(a(aab+a)+aa(a+ab))\}$; $R_2 = R_1^{-1} \cup \{(a(aab+a)+aa(a+ab), a(aab+a))\}$; $Q_1 = R_2$ y $Q_2 = Q_1^{-1} \cup \{(a(a+ab), aab+a), (a+ab, ab), (0, b)\}$.

Es fácil ver que $\mathcal{U}_f(\mathcal{G}_1) = \mathcal{G}_1$ y $\mathcal{U}_f(\mathcal{G}_2) = \mathcal{G}_2$, para todo $\mathcal{U}_f$, a lo sumo alguna de las etiquetas puede haber cambiado (a por a' ó b por b'). Por lo tanto $\mathcal{G}_1 \sim_2^{\mathbf{U}} \mathcal{G}_2$. Pero $\mathcal{G}_1 \not\rightleftharpoons_{R,2} \mathcal{G}_2$. Para demostrarlo basta con verificar que las relaciones R_i y Q_i que encontramos son únicas y que esas relaciones no son 2-nested ready similares: tomemos $0 S_2 b$ y veamos que los inicios de esos estados no son iguales: $I(0) = \emptyset \neq I(b) = b$.

Este ejemplo se puede generalizar de la forma en que se viene haciendo para obtener pares de STE U congruentes para todo n. Para ello se toma $\mathcal{G}_1^{n+1} = a.\mathcal{G}_2^n$ y $\mathcal{G}_2^{n+1} = a.\mathcal{G}_1^n + a.\mathcal{G}_2^n$.

Veamos ahora que $\rightleftharpoons_{n+1} \subseteq \rightleftharpoons_{R,n}$.

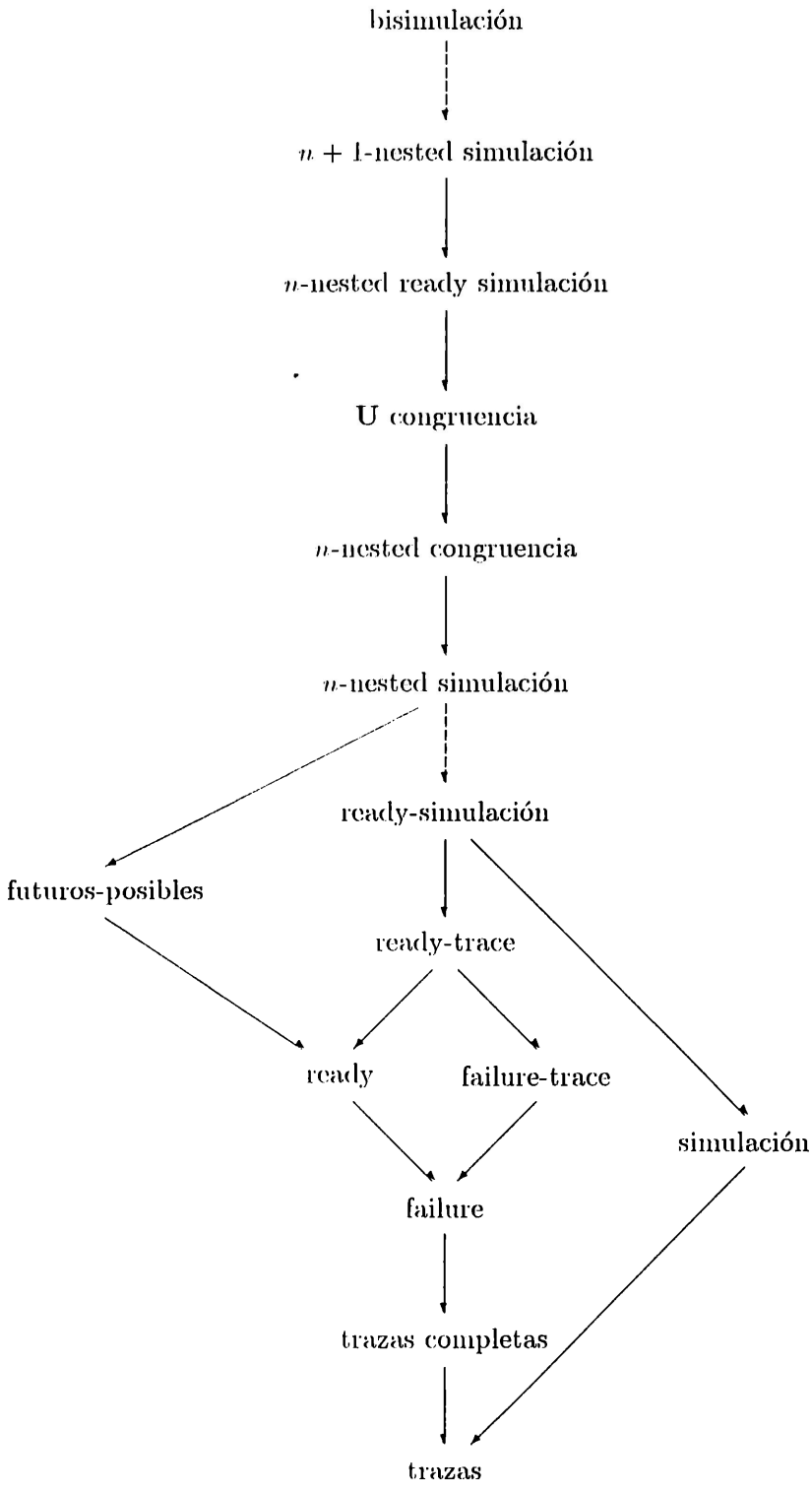


Figura 6.1: Reticulado de equivalencias semánticas

(**caso** $n = 2$) Sean $\mathcal{G}_1, \mathcal{G}_2$ STE tales que $R : \mathcal{G}_1 \subseteq_3 \mathcal{G}_2$. Por el teorema 5.3.10 sabemos que $forallf, R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_2 \mathcal{U}_f(\mathcal{G}_2)$ y, por definición de $\rightleftharpoons_n$, existe $Q : \mathcal{U}_f(\mathcal{G}_2) \subseteq_1 \mathcal{U}_f(\mathcal{G}_1)$, con $Q \subseteq R^{-1}$. En particular para $f = Id, R : \mathcal{G}_1 \subseteq_2 \mathcal{G}_2, Q : \mathcal{G}_2 \subseteq_1 \mathcal{G}_1$. Además, por el lema 5.3.9, $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_R \mathcal{U}_f(\mathcal{G}_2)$, con lo que R es una ready-simualción. Supongo que qQp. Si $a \in I(q)$, como Q es una simulación, entonces $a \in I(p)$. Si $a \in I(p)$, como $Q \subseteq R^{-1}$ sabemos que pRq y R es una simulación, entonces $a \in I(q)$. Así Q también es una ready-simulación. Finalmente $R : \mathcal{G}_1 \rightleftharpoons_{R,2} \mathcal{G}_2$.

(**caso** $n > 2$) Sean $\mathcal{G}_1, \mathcal{G}_2$ STE tales que $R : \mathcal{G}_1 \subseteq_{n+1} \mathcal{G}_2$. Por el teorema 5.3.10 sabemos que $forallf, R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_n \mathcal{U}_f(\mathcal{G}_2)$ y , por definición de $\rightleftharpoons_n$, existe $Q : \mathcal{U}_f(\mathcal{G}_2) \subseteq_{n-1} \mathcal{U}_f(\mathcal{G}_1)$, con $Q \subseteq R^{-1}$. En particular para $f = Id, R : \mathcal{G}_1 \subseteq_n \mathcal{G}_2$. Además, por el lema 5.3.9, $R : \mathcal{U}_f(\mathcal{G}_1) \subseteq_R \mathcal{U}_f(\mathcal{G}_2)$, con lo que R es una ready-simulación. Por hipótesis inductiva Q es una $n-1$-nested ready simulación y, finalmente, R es una n-nested ready simulación.

Volvamos ahora a los STE $\mathcal{G}_1^n$ y $\mathcal{G}_2^n$ de la nota 4.1.11 para probar que $\rightleftharpoons_{n+1} \neq \rightleftharpoons_{R,n}$. Los mismos son n-nested similares. En ambos podemos ver fácilmente que no van a ser afectados por ningún operador $\mathcal{U}_f$ (salvo que todas las acciones a cambien por a'), por lo tanto esos STE son U congruentes. Finalmente vemos que son n-nested ready similares ya que si tomamos p, q tales que pR_iq ó pQ_iq podemos ver que $I(p) = I(q)$.

Pero en esa misma nota vimos que $\mathcal{G}_1^n \not\rightleftharpoons_{n+1} \mathcal{G}_2^n$, con lo cual $\rightleftharpoons_{n+1} \neq \rightleftharpoons_{R,n}$. ■

Finalmente damos el nuevo reticulado de equivalencias semánticas (ver la figura 6.1) en donde incluimos las equivalencias nuevas. Note que $\rightleftharpoons_{n+1} \subset \rightleftharpoons_{R,n} \subset \sim_n^{\mathbf{U}} \subset \rightleftharpoons_n$.

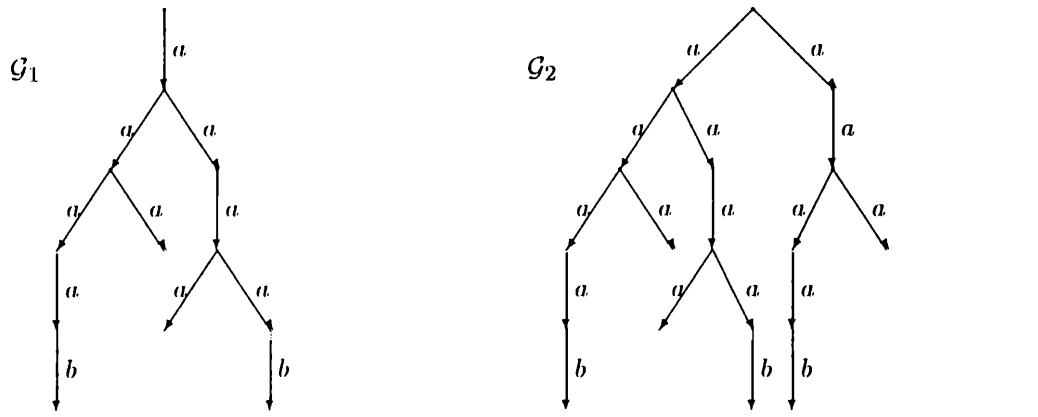


Figura 6.2: STE U congruentes para $n = 2$

Capítulo 7

Conclusiones

Un punto esencial en el uso de semántica de dos pasos para los sistemas reactivos es la compatibilidad entre los operadores y las equivalencias semánticas elegidas. Decimos que un operador es compatible con una equivalencia (o que la equivalencia es una congruencia para el operador) si el mismo preserva la equivalencia.

Este tipo de resultados permite describir los operadores directamente sobre los modelos concretos, lo que es en general simple, directo e intuitivo, y obtener, gratuitamente, la definición sobre la clase de equivalencia. Esto nos permite utilizar la equivalencia como equivalencia semántica.

Cuando una equivalencia no es una congruencia para un operador, es natural interrogarse sobre cual es la equivalencia más gruesa contenida en la anterior que si sea compatible.

En este trabajo se estudió la compatibilidad entre varios operadores sobre el modelo más difundido de Sistemas Reactivos, los Sistemas de Transiciones Etiquetadas [Kel76] (STE), y varias equivalencias sobre ellos que aparecen en la literatura.

Se comenzó trabajando con los operadores de prioridad [BBK85, BBK86, BW90] y luego se generalizaron los resultados a una clase amplia de operadores unarios sobre los STE. Se estudió el comportamiento respecto a estos operadores de una variedad de equivalencias propuestas en la literatura [Gla90]. En los casos en que estas equivalencias no resultaron congruencias, se planteó encontrar las mayores congruencias incluidas en ellas.

Los resultados obtenidos son:

Respecto a los operadores de prioridad, se demostró que son compatibles con la ready simulación, $\rightleftharpoons_R$. Que $\rightleftharpoons_R$ sea una congruencia para los operadores de prioridad es un resultado esperado a partir de los obtenidos para trazas y ready-traces en [BBK85].

Siguiendo esa misma intuición parecía evidente que $\rightleftharpoons_n$ también era una congruencia. Las n-nested simulaciones, $\rightleftharpoons_n$, son equivalencias estrictamente más finas que la ready simulación y por lo tanto la *información* con respecto a las *acciones iniciales* de un estado se encuentra disponible. Sin embargo en este trabajo se obtuvo que los operadores de prioridad no preservan las n-nested simulaciones, esto es, las $\rightleftharpoons_n$ no son congruencias para ellos.

Esto descarta la consideración de las $\rightleftharpoons_n$ como equivalencias semánticas, al menos en contextos en que los operadores de prioridad tengan sentido.

Para salvar este inconveniente se definió la *n-nested congruencia* (con respecto a θ), $\sim_n^\theta$, como la mayor congruencia para los operadores de prioridad contenida en $\rightleftharpoons_n$. Así se obtiene,

para cada n, una equivalenica semántica estrictamente más fina que la $\rightleftharpoons_n$ y estrictamente más gruesa que la bisimulación, $\leftrightarrows$, que es una congruencia para dichos combinadores (en realidad es más gruesa que $\rightleftharpoons_{n+1}$).

Esto nos permitió concluir que estos operadores no pueden ser dotados de una semántica operacional estructurada dentro del formato *tyft/tyxt*. Esto se debe a que todos los constructores a los que se le puede dar semántica en ese formato son compatibles con la 2-nested simulación, como se muestra en [GV89].

Más adelante se extendió la clase de los operadores estudiados para que contuviera a otros operadores comunes en la literatura, como son los de renombrado y ocultamiento. Dimos en llamar *clase* **U** a esa extensa clase de operadores estáticos unarios definidos sobre los STE. Se realizó, para cada equivalencia, el mismo tipo de análisis en busca de una equivalencia en la que pudieran tener sentido estos operadores.

Se demostró que la ready simulación y la bisimulación son congruencias para los operadores de **U**, afirmándose así que se está en presencia de una clase razonable de operadores.

Teniendo en cuenta que prioridad no es compatible con $\rightleftharpoons_n$ se obtiene que tampoco lo es la clase **U**. En este trabajo se demuestra que los operadores de **U** compatibles con esa equivalencia son exactamente los monótonos.

Tal como se vió para prioridad, a partir de [GV89], se obtiene que los operadores no monótonos de **U** no pueden ser dotados de una semántica operacional estructurada dentro del formato *tyft/tyxt*.

Se definieron además las n-nested congruencias con respecto a toda la clase **U**, ($\sim_n^{\mathbf{U}}$). Con ellas se obtuvo una familia infinita de equivalencias intercaladas entre las $\rightleftharpoons_n$, convergiendo en la bisimulación. En efecto, se demostró que la n-nested congruencia es estrictamente más fina que la $\rightleftharpoons_n$ y estrictamente más gruesa que la $\rightleftharpoons_{n+1}$.

Se buscó sin éxito una caracterización realcional de esas equivalencias, del tipo de las usadas en el capítulo 2. La búsqueda nos llevó hacia equivalencias más finas, en donde surgió la familia de las n-nested ready simulaciones, $\rightleftharpoons_{R,n}$. Esta familia infinita, también convergente a la bisimulación, resultó ser una congruencia para todos los operadores de **U**. Además fué posible darle una caracterización relacional. Así se obtuvo una familia infinita de equivalencias intercaladas entre las $\rightleftharpoons_n$ y la $\leftrightarrows$ que se puede usar en modelos en que los operadores de **U** tengan sentido.

El orden entre las equivalencias semánticas más finas mencionadas en este trabajo esta dado, para $n > 1$ por:

$$\leftrightarrows \subset \ldots \subset \rightleftharpoons_{n+1} \subset \rightleftharpoons_{R,n} \subset \sim_n^{\mathbf{U}} \subseteq \sim_n^{0} \subset \rightleftharpoons_n \subset \ldots$$
$$\ldots \subset \rightleftharpoons_{R,1} \equiv \rightleftharpoons_R \subset \sim_1^{\mathbf{U}} \subset \sim_1^{0} \subset \rightleftharpoons_1 \equiv \rightleftharpoons$$

Como trabajo futuro se presenta la posibilidad de obtener una caracterización lógica de $\rightleftharpoons_{R,n}$ basada en HML. También parece interesante el estudiar clases más amplias de operadores unarios sobre los STE, siguiendo dos objetivos: el primero es terminar de cubrir todos los combinadores naturales sobre los sistemas reactivos y, el segundo, obtener la clase de operadores para la cual la $\rightleftharpoons_{R,n}$ es la mayor congruencia contenida en $\rightleftharpoons_n$.

También se considera interesante incluir variantes de las equivalencias de [GV89] que contengan a la acción silenciosa τ, utilizada en sistemas reactivos. La acción τ permite

definir el operador de abstracción que no está incluído en nuestra clase U. Se encontró poca bibliografía con respecto a equivalencias que consideren τ y el operador de prioridad en un mismo contexto.

Agradecimientos

No quiero hacer enumeraciones sino que espero poder demostrar personalmente lo agradecida que estoy a toda la gente que ayudó a que este trabajo fuera posible. Gracias :)

Bibliografía

[BBK85] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Ready trace semantics for concrete process algebra with the priority operator. Reporte CS-R8517, Centrum voor Wiskunde en Informatica, Amsterdam, 1985.

[BBK86] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. In *Fundamentae Informaticae IX*, pages 127–168, 1986.

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984.

[BIM88] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced: preliminary report. In *Conference Record of the 15th ACM Symposium on Principles of Programming Languages*, pages 229–239, California, San Diego, 1988.

[BK85] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, (37):77–121, 1985.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process algebra.* Cambridge University Press, 1990.

[CH88] R. Cleaveland and M. Hennessy. Priorities in process algebras. In *Proceedings of LICS 88*, pages 193–202. IEEE, 1988.

[CN94] R. Cleaveland and V. Natarajan. Priority and abstraction in process algebra, 1994. Submitted for publication.

[DEP94a] P.R. D'Argenio, J.V. Echague, and C. Pertino. Congruence results for a large family of static unary operators. In C. Isaac and R. Peralta, editors, *Proocedings of XIV Conferencia Internacional de la SCCC*, pages 337–348, 1994.

[DEP94b] P.R. D'Argenio, J.V. Echague, and C. Pertino. The n-nested simulation is not a congruence for the priority operators. In *Proocedings of XX Conferencia Latinoamericana de Informática*, pages 849–872. Noriega Editores, 1994.

[Gla90] R.J. van Glabbeek. The linear time - branching time spectrum. Reporte CS-R9029, Centrum voor Wiskunde en Informatica, Amsterdam, 1990.

[Gla93] R.J. van Glabbeek. The linear time - branching time spectrum II. The semantics of sequential systems with silent moves (preliminary version), 1993.

[GV89] J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence (extended abstract). In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, editors, *Proceedings ICALP 89*, pages 423–438, Stressa, 1989. LNCS 372, Springer-Verlag.

[Hen90] M. Hennessy. *The semantics of programming languages: an elementary introduction using structural operational semantics.* Jhon Wiley & Sons, Sussex, 1990.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137 161, 1985.

[Hoa85] C.A.R. Hoare. *Communicating sequential process.* Prentice Hall, 1985.

[Kel76] R.M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 8(19):371 384, 1976.

[Mil80] R. Milner. *A calculus of communicating systems*, volume 92 of *LNCS*. Springer-Verlag, 1980.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, (25):267-310, 1983.

[Mil89] R. Milner. *Communication and concurrency.* Prentice Hall, 1989.

[OH83] E.-R. Olderog and C.A.R. Hoare. Specification-oriented semantics for communicating processes. In J. Diaz, editor, *Proceedings 10th ICALP*, pages 561–572, Barcelona, 1983. LNCS 154, Springer-Verlag.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequence. In P. Deussen, editor, *Proceedings 5th. GI Conference*, pages 167–183. LNCS 104, Springer-Verlag, 1981.

[Phi87] I.C.C. Phillips. Refusal testing, 1987.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Reporte DAIMI-FN-19, Computer Science Department, University of Århus, 1981.

[Pnu85] A. Pnueli. Linear and branching structures in the semantic and logic of the reactive systems. In W. Brauer, editor, *Proceedings 12th ICALP*, pages 15–32, Nafplion, 1985. LNCS 194, Springer-Verlag.

[RS81] W.C. Rounds and Brookes S.D. Possible futures, acceptances, refusals and communicating processes. In P. Deussen, editor, *Proceedings 22nd Annual Symposium on Foundations of Computer Science*, pages 140–149. IEEE, 1981.